Oracle Database Express Edition®

2 Day Plus PHP Developer Guide

10g Release 2 (10.2)

B25317-01

September 2005

Oracle Database Express Edition 2 Day Plus PHP Developer Guide, 10g Release 2 (10.2)

B25317-01

Contributing Author: Christopher Jones, Simon Law, Glenn Stokol, Simon Watt.

Contents

Preface

The *Oracle Database Express Edition 2 Day Plus PHP Developer Guide* introduces developers to the use of PHP to access Oracle Database Express Edition.

This preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

The *Oracle Database Express Edition 2 Day Plus PHP Developer Guide* is intended as an introduction to application development using Zend Core for Oracle and Oracle Database Express Edition.

This document assumes a basic understanding of the SQL, PL/SQL and PHP.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Related Documents

For more information, see these Oracle resources:

- *Oracle Database Express Edition Installation and Licensing Guide for Linux*
- *Oracle Database Express Edition 2 Day DBA Guide*
- *Oracle Database Express Edition 2 Day Developer Guide*
- *Oracle HTML DB User's Guide*
- *Oracle HTML DB 2 Day Developer*
- *Oracle Database Express Edition 2 Day Plus Java Developer Guide*
- *Oracle Database Express Edition 2 Day Plus .NET Developer Guide*
- *Oracle Database Express Edition ISV Embedding Guide*
- *SQL*Plus User's Guide and Reference*
- *SQL*Plus Quick Reference*
- *Oracle Database PL/SQL User's Guide and Reference*
- *Oracle Database SQL Reference*
- Oracle Call Interface Programmer's Guide
- *Oracle Database Concepts*
- *Oracle Database Application Developer's Guide - Fundamentals*
- *Oracle Database Globalization Support Guide*
- *Oracle Database Error Messages*

The examples in this book use the HR sample schema, which is installed by default. See *Oracle Database Sample Schemas* for information about this schema.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introducing PHP with Oracle Database XE

Oracle® Database Express Edition (Oracle Database XE) is a relational database that you can use to store, use, and modify data. Zend Core for Oracle enables application development using PHP.

This chapter has the following topics:

Zend Core for Oracle

Zend Core for Oracle, developed in partnership with Zend Technologies provides a seamless out-of-the-box experience delivering a stable, high performance, easy-to-install and supported PHP development and production environment fully integrated with Oracle Database Express Edition.

Purpose

This guide is a tutorial that shows you how to use Zend Core for Oracle to connect to Oracle Database XE, and demonstrates how to use PHP to access and modify data.

Overview of the Sample Application

This document guides you through the development of a simple Human Resources (HR) application for a fictitious company ""AnyCo Corp".

The application manages departmental and employee information stored in the `DEPARTMENTS` and `EMPLOYEES` tables in the HR schema provided with Oracle Database XE.

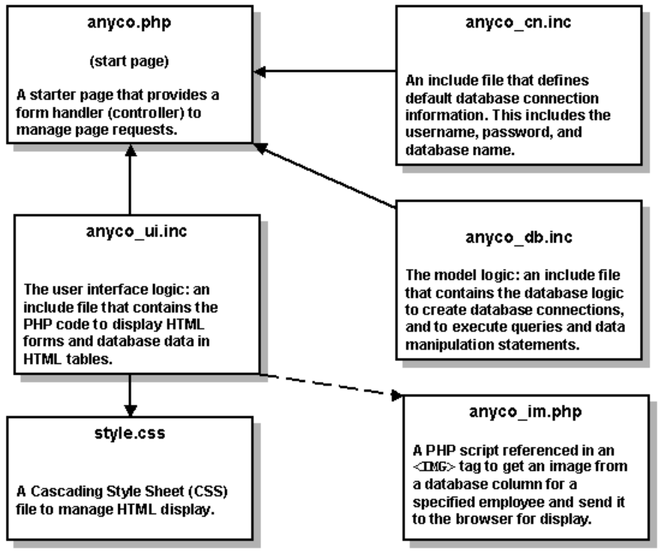
The complete sample application:

1. Establishes a connection to the database using PHP's OCI8 extension
2. Queries the database for department and employee data
3. Displays and navigates through the data
4. Shows how to insert, update, anddelete employee records
5. Handles data exceptions

6. Uploads and displays employee photographs

Figure 1-1 shows the relationship between the files developed for this application:

Figure 1–1 Functionality in the Sample PHP Application


anyco.php
(start page)
A starter page that provides a form handler (controller) to manage page requests.
anyco_cn.inc
An include file that defines default database connection information. This includes the username, password, and database name.
anyco_ui.inc
The user interface logic: an include file that contains the PHP code to display HTML forms and database data in HTML tables.
anyco_db.inc
The model logic: an include file that contains the database logic to create database connections, and to execute queries and data manipulation statements.
style.css
A Cascading Style Sheet (CSS) file to manage HTML display.
anyco_im.php
A PHP script referenced in an <IMG> tag to get an image from a database column for a specified employee and send it to the browser for display.

The sample application files are:

anyco.php
This file has the main logic for the AnyCo application. It contains control logic to decide which page is displayed. It manages session data for navigation. The functionality in `anyco_cn.inc`, `anyco_db.inc` , and `anyco_ui.inc` is used by it.

anyco_ui.inc
This file contains the functions used for presentation of data and forms in a HTML page.

anyco_cn.inc
This file contains definitions for database connection information: the database username, password, and database connect identifier.

anyco_db.inc
This file contains database logic to create connections, execute queries, and execute data manipulation statements.

anyco_im.php
This file contains logic to retrieve an image from a database column and send it to a browser for display as a JPEG image.

style.css
This file contains Cascading Style Sheet definitions for various HTML tags generated by the application. It manages the look and feel of the application.

Files with the suffix `.inc` are PHP code files included in other PHP files.

Files with the suffix `.php` can be loaded in a browser.

You can create and edit the PHP application source files in a text editor or any tool that supports PHP development.

The code for each chapter builds on the files completed in the previous chapter.

This tutorial creates files in your `$HOME/public_html` directory. This is the default location for web access if the Apache web server configuration has the `UserDir` directive enabled. If you create files in a different location, you need to change the steps for file editing and execution to match your working directory name and URL.

Resources

- Oracle Database Express Edition Developer Center on the Oracle Technology Network at: `http://www.oracle.com/technology/xe`
- PHP Developer Center on the Oracle Technology Network at: `http://www.oracle.com/technology/tech/php/index.html`
- Zend Core for Oracle Developer Center at: `http://www.oracle.com/technology/tech/php/zendcore/index.html`
- Oracle Database Express Edition documentation on the Oracle Technology Network at: `http://www.oracle.com/technology/xe/documentation`
- The Oracle Database Documentation Library on the Oracle Technology Network at: `http://www.oracle.com/technology/documentation`

2 Getting Started

This chapter explains how to install and test your Oracle Database Express Edition (Oracle Database XE) and PHP environment. It has the following topics:

What You Need

- Oracle Database Express Edition 10*g*R2
- Apache 1.3.x or later
- Zend Core for Oracle
- Text editor for editing PHP code.

Obtaining Oracle Database Express Edition (Oracle Database XE)

Oracle Database Express Edition is available from the Oracle Technology Network at: `http://www.oracle.com/technology/xe.`

See also:

- Oracle Database XE discussion Forum at: `http://www.oracle.com/technology/xe/forum`
- Oracle Database XE documentation at: `http://www.oracle.com/technology/xe/documentation`

Obtaining Apache

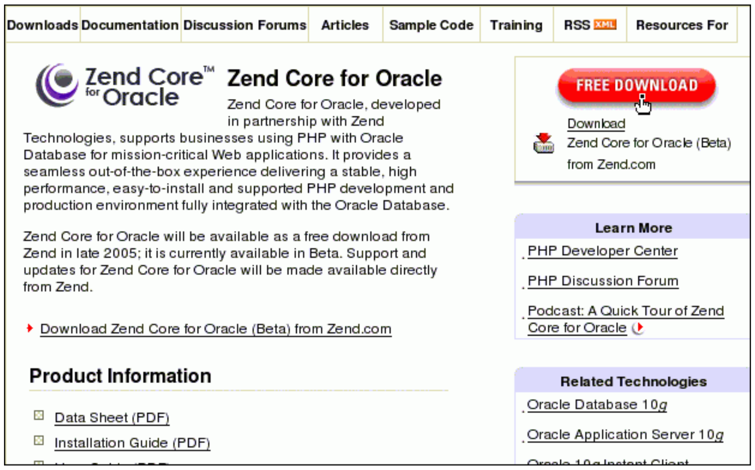
Apache is normally a standard part of the Linux environment. If Apache is not available on your Linux platform, you can download it from `http://httpd.apache.org`

Obtaining Zend Core for Oracle

1. To obtain Zend Core for Oracle for the Linux Platform enter the following URL in your Web Browser:

http://www.oracle.com/technology/tech/php/zendcore/index.html

2. To the right of the "Zend Core for Oracle" Web page, click the **Free Download** image:



Zend Core for Oracle, developed in partnership with Zend Technologies, supports businesses using PHP with Oracle Database for mission-critical Web applications. It provides a seamless out-of-the-box experience delivering a stable, high performance, easy-to-install and supported PHP development and production environment fully integrated with the Oracle Database.

Zend Core for Oracle will be available as a free download from Zend in late 2005; it is currently available in Beta. Support and updates for Zend Core for Oracle will be made available directly from Zend.

Download Zend Core for Oracle (Beta) from Zend.com

Download Zend Core for Oracle (Beta) from Zend.com

3. Save the downloaded file in a temporary directory, such as /tmp.

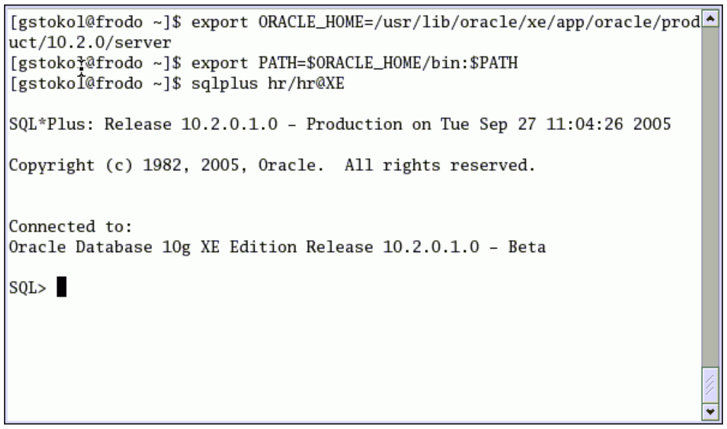
Testing the Oracle Database XE Installation

1. The PHP application connects to the database as the HR user. You may need to execute the following SQL command, as a user with DBA privileges:

```
alter user hr account unlock identified by hr;
```

2. To test that Oracle Database XE is accessible, connect to the database using the HTML DB web interface, or enter the following commands in a command window:

```
export ORACLE_HOME=
export PATH=$ORACLE_HOME/bin:$PATH

sqlplus hr/hr@localhost
```



```
[gstokol@frodo ~]$ export ORACLE_HOME=/usr/lib/oracle/xe/app/oracle/prod
uct/10.2.0/server
[gstokol@frodo ~]$ export PATH=$ORACLE_HOME/bin:$PATH
[gstokol@frodo ~]$ sqlplus hr/hr@XE

SQL*Plus: Release 10.2.0.1.0 - Production on Tue Sep 27 11:04:26 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g XE Edition Release 10.2.0.1.0 - Beta

SQL>
```

Terminate the SQL*Plus session by entering the exit command at the SQL prompt:

```
SQL> exit
```

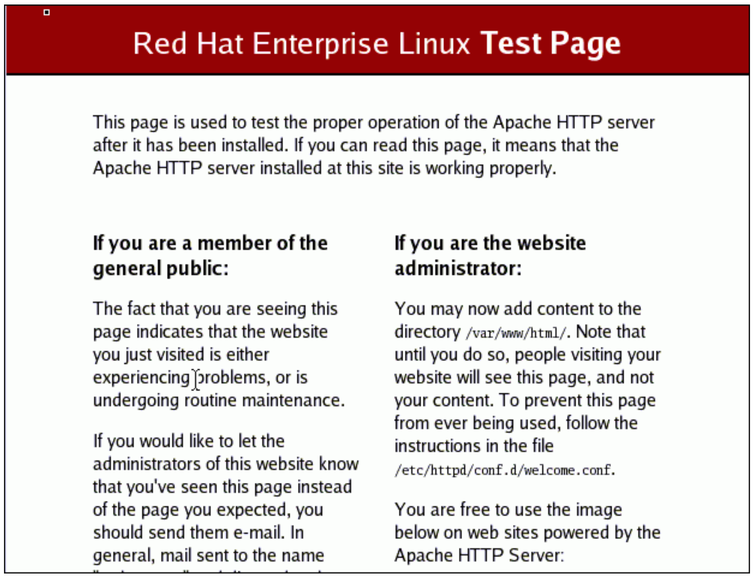
See also: For further information about unlocking an Oracle Database account, see Chapter 6, "Managing Users and Security", in the *Oracle Database Express Edition 2 Day DBA* guide.

Testing the Apache Installation

1. Start your web browser and enter the following URL:

   ```
   http://localhost
   ```

   Your browser should display a page similar to:



Red Hat Enterprise Linux **Test Page**

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the Apache HTTP server installed at this site is working properly.

If you are a member of the general public:

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name

If you are the website administrator:

You may now add content to the directory /var/www/html/. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file /etc/httpd/conf.d/welcome.conf.

You are free to use the image below on web sites powered by the Apache HTTP Server:

2. In the default Apache Web server configuration file set up a public virtual directory as public_html for accessing your PHP files. By using your preferred editor open the Apache configuration file `/etc/httpd/conf/httpd.conf` and remove the "#" character at the start of the line with the following directive:

   ```
   #UserDir public_html
   ```

   This enables a browser to make a HTTP request using a registered user on the system and to serve files from the users `$HOME/public_html` directory. For example:

   ```
   http://localhost/~user
   ```

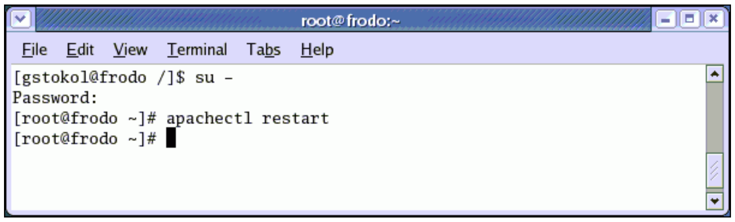
   For example: your Apache `httpd.conf` file should contain the following lines:

   ```
   <IfModule mod_userdir.c>
       #
       # UserDir is disabled by default since it can confirm the presence
       # of a username on the system (depending on home directory
       # permissions).
       #
       #UserDir disable

       #
       # To enable requests to /~user/ to serve the user's public_html
       # directory, remove the "UserDir disable" line above, and uncomment
   ```

```
    # the following line instead:
    #
    UserDir public_html
</IfModule>
```

3. In a command window, to use the new Apache configuration file restart Apache by entering the following commands:

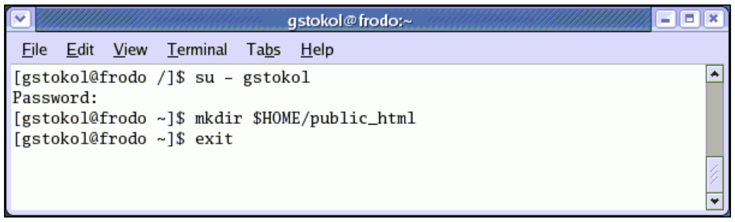
   ```
   su
   Password: <enter your su (root) password>
   apachectl restart
   ```



```
[gstokol@frodo /]$ su -
Password:
[root@frodo ~]# apachectl restart
[root@frodo ~]#
```

4. In the command window, login as a normal (non-root) user and create a `public_html` sub-directory in the users `$HOME` directory, by using the following commands:

   ```
   su - gstokol
   Password for gstokol: <enter the password>
   mkdir $HOME/public_html
   ```



```
[gstokol@frodo /]$ su - gstokol
Password:
[gstokol@frodo ~]$ mkdir $HOME/public_html
[gstokol@frodo ~]$ exit
```

5. If Apache is not running and you get an error page or do not get the expected results. In a command window, start the Apache Web server on your machine using the following commands:

   ```
   su
   Password: <enter your su (root) password>
   apachectl start
   ```

   If the Apache Web server does not start you may need to check the error log files to determine the cause. It may be a configuration error.

Setting Up Zend Core for Oracle

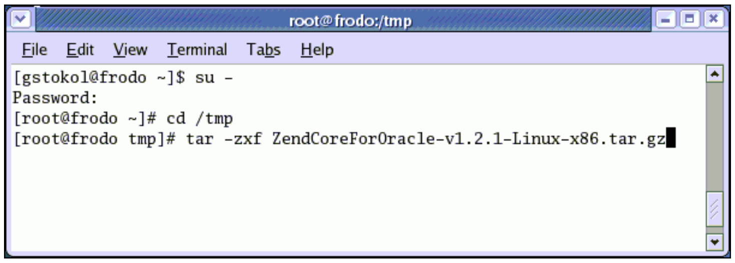
This tutorial is specific to PHP in Zend Core for Oracle.

For detailed setup information for Zend Core for Oracle, see the Installation Guide under Product Information on the Zend Core for Oracle web page at `http://www.oracle.com/technology/tech/php/zendcore/index.html`.

Installing Zend Core for Oracle on Linux

1. To extract the contents of the downloaded Zend Core for Oracle software as a root user, which is required to install the software. In a command window enter:

```
su -
Password: <enter the root password>
cd /tmp
tar -zxf ZendCoreForOracle-v1.2.1-Linux-x86.tar.gz
```

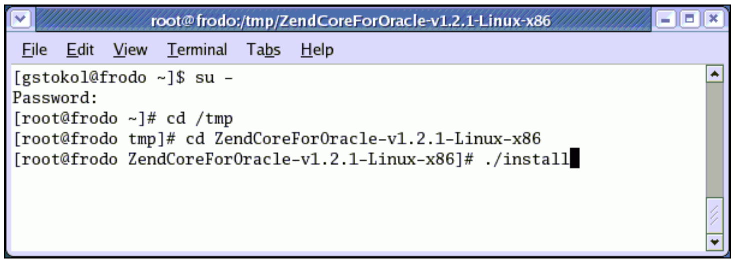


```
[gstokol@frodo ~]$ su -
Password:
[root@frodo ~]# cd /tmp
[root@frodo tmp]# tar -zxf ZendCoreForOracle-v1.2.1-Linux-x86.tar.gz
```

By default, unless specified otherwise, files are extracted to a sub-directory called `ZendCoreForOracle-v1.2.1-Linux-x86`.

2. To start the Zend Core for Oracle installation process, enter the following commands:

```
cd ZendCoreForOracle-v1.2.1-Linux-x86
./install
```


root@frodo:/tmp/ZendCoreForOracle-v1.2.1-Linux-x86
File Edit View Terminal Tabs Help

```
[gstokol@frodo ~]$ su -
Password:
[root@frodo ~]# cd /tmp
[root@frodo tmp]# cd ZendCoreForOracle-v1.2.1-Linux-x86
[root@frodo ZendCoreForOracle-v1.2.1-Linux-x86]# ./install
```

The install command must be executed with root user privileges. After the `./install` command is entered the installation process begins as documented in subsequent steps.

3. In the initial "Zend Core for Oracle Installation" page, click **OK**.

4. In the "Zend Core for Oracle V.1" page, read the license agreement. To continue with the installation, click **Exit**.

5. When prompted to accept the terms of the license, click **Yes**.

6. When prompted to specify the location for installing Zend Core for Oracle, accept the default (or enter your preferred location), and click **OK**.

   The installer begins extracting the files required for the installation.

7. When the progress window indicates all the software has been installed you are prompted to "Please enter the GUI password". In the "Password" field, enter your password, for example `oracle`, and click **OK**.

   The password specified here allows you to log into the Zend Core for Oracle administration Web pages to enable configuration of Zend Core for Oracle engine directives or property values.

8. When prompted to "Verify the password", enter the same password as specified in the previous step and click **OK**.
9. In the Zend Core support page, you may optionally enter a Zend network user ID and password. In this case, the assumption is that you have already registered a Zend network user ID and password when you downloaded the software, and therefore click **No**.

   If you have not registered, you may still click No, and register at a later time using the Zend Core Web page `http://www.zend.com`.
10. The next page prompts you to select the web server for Zend Core installation. Select the first entry **Apache 2.0.52 (/etc/httpd/conf/httpd.conf)**, the default Apache installed with Linux. Click **OK**.

    If you desire, you may continue to install Zend Core with other supported Web servers installed on your system.
11. In the page confirming your Web server selection, at the "Do you wish to proceed?" prompt, click **Yes**.
12. In the next installation page, you are prompted to "Please select an installation method for Apache 2.0.52". Select the first entry **Apache module** as the method, and click **OK**.
13. In the next installation page, prompting "Please select a virtual server for the Zend Core GUI", select the first entry **Main Server**, click **OK**.
14. In the next installation page, after selecting the virtual server, at the "Would you like to restart the Web Server" prompt, click **Yes**.
15. When prompted "Would you like to configure another Web Server to use Zend Core", click **No**.
16. In the final installation page containing "Thank you for installing Zend Core for Oracle" lists useful configuration commands and Web page for administration of the Zend Core engine. Take note of the information and click **EXIT**.
17. When the Zend Core installation pages are terminated, a message is displayed in your command window.

    The Zend Core for Oracle installation is now complete.

Configuring Zend Core for Oracle

In this section, you configure environment variables, and Zend Core directives that control default error reporting in web pages.

1. In a web browser, enter the following URL to access the Zend Core administration page:

   ```
   http://localhost/ZendCore
   ```

   The Zend Core for Oracle welcome page is displayed.
2. In the Zend Core for Oracle Welcome page in the Password field, enter the GUI password, which you provided during Zend Core for Oracle installation. Click the login >>> icon.
3. In the Zend Core for Oracle administration GUI page, the main "Control Center" tab page is displayed with the "System Overview" tab page selected. To display the configuration options, click the **Configuration** tab.

4. In the PHP tab page, which is selected by default, expand the Error Handling and Logging configuration entry by clicking the **+** icon.
5. In the PHP Configuration page, to enable the display of errors in the HTML script output, set the `display_errors` directive **On**.

   The GUI application is aware that you have unsaved changes. Under the PHP Configuration page header notice the "Unsaved configuration" message.
6. In the PHP Configuration page, to save the configuration change, click the **Save Settings** link.

   Saving configuration changes typically requires the Apache server to be restarted. Under the PHP Configuration page header notice the "Please Restart Apache" message.
7. In the PHP Configuration page, to restart the Apache server click the **Restart Server** link.

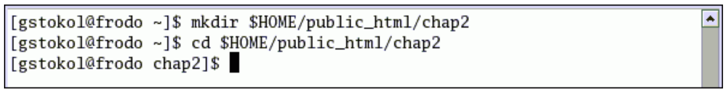
   The PHP Configuration page is refreshed when the Apache server has been restarted.
8. In the Zend Core for Oracle administration page, to exit the GUI page click the **Logout** link.

   Now that the basic configuration changes have been made, you may now proceed to the next section to test the Zend Core for Oracle installation.

Testing the Zend Core for Oracle Installation

1. To get started, create a directory called `chap2` as a child of your `$HOME/public_html` directory and change directory to `$HOME/public_html/chap2` by entering the following commands:

```
mkdir $HOME/public_html/chap2
cd $HOME/public_html/chap2
```



```
[gstokol@frodo ~]$ mkdir $HOME/public_html/chap2
[gstokol@frodo ~]$ cd $HOME/public_html/chap2
[gstokol@frodo chap2]$
```

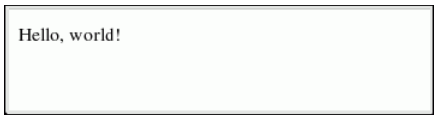
2. To check that PHP works, with your preferred editor, create a file called `hello.php` which contains the following HTML text:

```
<?php
  echo "Hello, world!";
?>
```

3. Open a web browser and enter the following URL:

```
http://localhost/~<username>/chap2/hello.php
```

   The result in the browser is:



```
Hello, world!
```

3 Getting Connected

In this chapter you create HR application files which implement PHP functions to connect and disconnect with the Oracle Database. You also develop a PHP function which enables you to execute a query to validate that a database connection has been successfully established.

It guides you through the creation and modification of PHP files that call a function to produce the header and footer for the HR application report pages, where the footer section of the page includes a date and time.

This chapter has the following topics:

Note: For simplicity, the username and password are written into this sample application code. For applications that will be deployed, coding the username and password strings directly into your application source code is not recommended. Some other technique, such as implementing a dialog that prompts the end user for the username and password is recommended.

See the *Oracle Database Security Guide* and documentation for your development environment for details on security features and practices.

Building the Departments Page

1. To create a directory for the application files, in a command window enter the following command:

   ```
   mkdir $HOME/public_html/chap3
   cd $HOME/public_html/chap3
   ```

2. To start creating the PHP application user interface framework create a file called `anyco_ui.inc` that contains the two functions `ui_print_header()` and `ui_print_footer()` with their parameters to enable web pages with consistent header and footer sections:

   ```
   <?php

   function ui_print_header($title)
   {
   ```

```
  $title = htmlentities($title);
  echo <<<END
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
     "http://www.w3.org/TR/html4/strict.dtd">
  <html>
  <head>
    <meta http-equiv="Content-Type"
          content="text/html; charset=ISO-8859-1">
    <link rel="stylesheet" type="text/css" href="style.css">
    <title>Any Co.: $title</title>
  </head>
  <body>
  <h1>$title</h1>
END;
}

function ui_print_footer($date)
{
  $date = htmlentities($date);
  echo <<<END
  <div class="footer">
    <div class="date">$date</div>
    <div class="company">Any Co.</div>
  </div>
END;
}

?>
```

- The design of this application makes use of PHP function definitions to enable modular reusable code.
- The PHP functions defined in the `anyco_ui.inc` file contain parts of the original HTML contents from the first `anyco.php` you created.
- The functions in `anyco_ui.inc`, make use of a PHP language construct called a "here document". This enables you to place any amount of HTML formatted text between the following two lines:

  ```
  echo <<<END
  END;
  ```

  The `END;` line must not be prefixed with leading spaces otherwise the rest of the document is treated as part of the text to be printed. Any PHP parameters appearing inside the body of a "here document" are replaced with their values, for example the `$title` or `$date` parameters.
- The PHP function `htmlentities()` is used to prevent user-supplied text from containing HTML markup.

3. The PHP file makes use of a Cascading Style Sheet (CSS) file called `style.css` to specify presentation style in HTML in the browser.

Use your editor to create `style.css` in the `chap3` directory with the following CSS text:

```
body
{ background: #CCCCFF;
  color:      #000000;
  font-family: Arial, sans-serif; }

h1
```

```
{ border-bottom: solid #334B66 4px;
  font-size: 160%; }

table
{ padding: 5px; }

td
{ border: solid #000000 1px;
  text-align: left;
  padding: 5px; }

th
{ text-align: left;
  padding: 5px; }

.footer
{ border-top: solid #334B66 4px;
  font-size: 90%; }

.company
{ padding-top: 5px;
  float: right; }

.date
{ padding-top: 5px;
  float: left; }
```

4. To call the user interface functions create `anyco.php` with the following text:

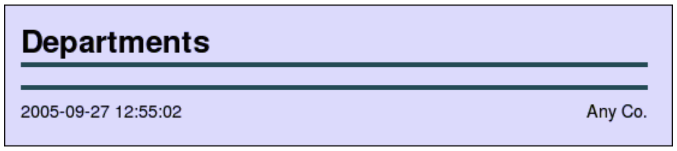
```
<?php

require('anyco_ui.inc');

ui_print_header('Departments');
ui_print_footer(date('Y-m-d H:i:s'));

?>
```

   The `anyco.php` file uses PHP functions to produce HTML content. The `require()` PHP function is used to include the code in `anyco_ui.inc`, such that the functions defined in it can be called to produce the desired result. If the required file cannot be found, PHP will generate an error and stop running the script.

5. To test `anyco.php`, enter the following URL in your browser:

```
http://localhost/~<username>/chap3/anyco.php
```

The resulting Web page produced is:



Departments

2005-09-27 12:55:02 Any Co.

The date and time appear in the page footer section.

Connecting to the Database

1. To form a database connection in your PHP application, you use the `oci_connect()` function with three string parameters:

   ```
   $conn = oci_connect($username, $password, $db)
   ```

   The first and second parameters are the database username and password, respectively. The third parameter is the database connection identifier. The `oci_connect()` function returns a connection resource needed for other OCI8 calls, otherwise it returns FALSE if an error occurs. The connection identifier return is stored in a variable called `$conn`.

   To validate that the `oci_connect()` call returns a usable database connection, write a `do_query()` function that accepts two parameters: the database connection identifier, obtained from the call to `oci_connect()`, and a query string to select all the rows from the DEPARTMENTS table.

   Edit `anyco.php` to form a database connection with the following parameter values:

   - Username is `hr`.
   - Password for this example is `hr`. Remember to use the actual password of your `HR` user.
   - Oracle connect identifier is `//localhost/XE`.

   The file becomes:

   ```
   <?php // File: anyco.php

   require('anyco_ui.inc');

   // Create a database connection
   $conn = oci_connect('hr', 'hr', '//localhost/XE');

   ui_print_header('Departments');
   do_query($conn, 'SELECT * FROM DEPARTMENTS');
   ui_print_footer(date('Y-m-d H:i:s'));

   // Execute query and display results
   function do_query($conn, $query)
   {
     $stid = oci_parse($conn, $query);
     $r = oci_execute($stid, OCI_DEFAULT);

     print '<table border="1">';
     while ($row = oci_fetch_array($stid, OCI_RETURN_NULLS)) {
       print '<tr>';
       foreach ($row as $item) {
         print '<td>'.
               ($item ? htmlentities($item) : '&nbsp;').'</td>';
       }
       print '</tr>';
     }
     print '</table>';
   }

   ?>
   ```

The `oci_parse()` function prepares the query for execution, and is supplied the connection identifier and query string as the first and second parameters, respectively. The `oci_parse()` function returns a statement identifier needed to execute the query and fetch the resulting data rows, otherwise it returns FALSE on error.

The `oci_execute()` function executes the statement associated with the statement identifier provided in the first parameter. The second parameter specifies the execution mode. `OCI_DEFAULT` is used to indicate you do not want to statements to be committed automatically. The default execution mode is `OCI_COMMIT_ON_SUCCESS`. The `oci_execute()` function returns `TRUE` on success, otherwise it returns `FALSE`.

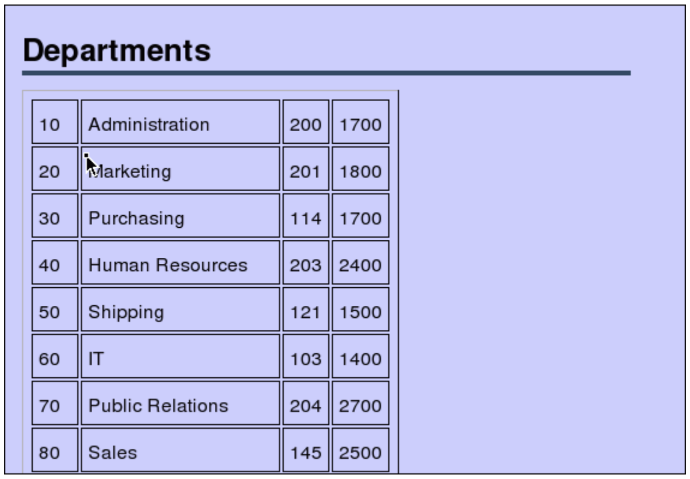
A while loop is used to fetch all the rows for the query executed. The `oci_fetch_array()` returns the next row from the result data, otherwise it returns `FALSE` if there are no more rows. The second parameter to `oci_fetch_array()` of `OCI_RETURN_NULLS` indicates that `NULL` database fields will be returned as PHP NULL values.

Each row of data is return as an associative or numeric array of column values. The code uses a PHP `foreach` construct to loop through the array and print each column value in a HTML table cell, inside a table row element. If the item value is `NULL` then a non-breaking space is printed, otherwise the item value is printed.

2. To test the changes made to `$HOME/public_html/chap3/anyco.php`, save the modified `anyco.php` file and in a browser window enter the following URL:

```
http://localhost/~<username>/chap3/anyco.php
```

The page returned in the browser window should resemble the following page:



Departments

10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500

If you wanted to query the EMPLOYEES data you could change the query in the `do_query()` function call to:

```
do_query($conn, 'SELECT * FROM EMPLOYEES');
```

Other Ways to Connect

In some applications using a persistent connection improves performance by removing the need to reconnect each time the script is called. Depending on your Apache configuration, this may cause a number of database connections to remain open simultaneously. The connection performance benefits need to be balanced with the overhead on the database server.

Persistent connections are made with the OCI8 `oci_pconnect()` function. The lifetime of persistent connections can be controlled by several settings in the PHP initialization file. Some the settings include:

oci8.max_persistent - controls the number of persistent connections per process.

oci8.persistent_timeout - specifies the time (in seconds) that a process maintains an idle persistent connection.

oci8.ping_interval - specifies the time (in seconds) that must pass before a persistent connection is "pinged" to check its validity.

See the PHP reference manual `http://www.php.net/manual/en/ref.oci8.php` for more information.

Disconnecting from the Database

The PHP engine will automatically close the database connection at the end of the script unless a persistent connection was made. To explicitly close a database connection you may call the `oci_close()` OCI function with the connection identifier returned by the `oci_connect()` call. For example:

```
<?php

$conn = oci_connect('hr', 'hr', '//localhost/XE');
...
oci_close($conn);

...

?>
```

4 Querying Data

In this chapter you extend the Anyco HR application from chapter 3 by adding additional information to the departments form. You also implement the functionality to query, insert, update, and delete employees in a specific department.

This chapter has the following topics:

Centralizing the Database Application Logic

Modify your application code by moving the database access logic into separate files for inclusion in the PHP application. Create new files in the `$HOME/public_html/chap4` directory.

1. Copy the files completed in chapter 3, to a new `chap4` directory:

```
mkdir $HOME/public_html/chap4
cp $HOME/public_html/chap3/* $HOME/public_html/chap4
cd $HOME/public_html/chap4
```

2. Using your preferred editor, create a file called `anyco_cn.inc`, which defines named constants for the database connection information. This file enables to change connection information in one place.

```
<?php // File: anyco_cn.inc

define('ORA_CON_UN', 'hr');             // Username
define('ORA_CON_PW', 'hr');             // Password
define('ORA_CON_DB', '//localhost/XE'); // Connection identifier

?>
```

3. Create a file called `anyco_db.inc` that declares functions for creating a database connection, executing a query, and disconnecting from the database. Use the following logic, which includes some error handling that is managed by calling an additional function called `db_error ()`:

```
<?php  // File: anyco_db.inc
```

```
function db_connect()
{
  // use constants defined in anyco_cn.inc
  $conn = oci_connect(ORA_CON_UN, ORA_CON_PW, ORA_CON_DB);
  if (!$conn) {
    db_error(null, __FILE__, __LINE__);
  }
  return($conn);
}

function db_do_query($conn, $statement)
{
  $stid = oci_parse($conn, $statement);
  if (!$stid) {
    db_error($conn, __FILE__, __LINE__);
  }

  $r = oci_execute($stid, OCI_DEFAULT);
  if (!$r) {
    db_error($stid, __FILE__, __LINE__);
  }
 $r = oci_fetch_all($stid, $results, null, null,
                    OCI_FETCHSTATEMENT_BY_ROW);
  return($results);
}

// $r is the resource containing the error.
// Pass no argument or false for connection errors
function db_error($r = false, $file, $line)
{
  $err =  $r ? oci_error($r) : oci_error();

  if (isset($err['message'])) {
    $m = htmlentities($err['message']);
  }
  else {
    $m = 'Unknown DB error';
  }

  echo '<p><b>Error</b>: at line '.$line.' of '.$file.'</p>';
  echo '<pre>'.$m.'</pre>';

  exit;
}

?>
```

The `db_do_query()` has been written to use the `oci_fetch_all()` OCI8 function, instead of `oci_fetch_array()`. The `oci_fetch_all()` function accepts five parameters.

- `$stid`, the statement identifier for the statement executed
- `$results`, the output array variable containing the data returned for the query
- The `null` in the third parameter for the number of initial rows to skip is ignored.

- The `null` in the fourth parameter for the maximum number of rows to fetch is ignored. In this case, all the rows for the query are returned. For this example where the result set is not large, it is acceptable.
- The last parameter flag `OCI_FETCHSTATEMENT_BY_ROW` indicates that the data in the `$results` array is organized by row, where each row contains an array of column values. A value of `OCI_FETCHSTATEMENT_BY_COLUMN` causes the `results` array to be organized by column, where each column entry contains an array of column values for each row. Your choice of value for this flag depends on how you intend to process the data in your logic.

To examine the structure of the result array use the PHP `var_dump()` function after the query has been executed. This is useful for debugging. For example:

```
print '<pre>';
var_dump($results);
print '</pre>';
```

The `db_error()` function, accepts three arguments. The `$r` parameter can be false or null for obtaining connection errors, or a connection resource or statement resource to obtain an error for those contexts. The `$file` and `$line` values are populated by using `__FILE__` and `__LINE__` respectively as the actual parameters to enable the error message to display the source file and line from which the database error is reported. This enables you to easily track the possible cause of errors.

The db_ `error()` function calls the `oci_error()` function to obtain database error messages.

The `db_error()` function calls `isset()` function checks if the message component of the database error structure is set before printing the message, or indicating that the error is unknown.

4. Edit `anyco_ui.inc`. To format the results of single row from the DEPARTMENTS table query in a HTML table format, insert the following function:

```
function ui_print_department($dept)
{
  if (!$dept) {
    echo '<p>No Department found</p>';
  }
  else {
    echo <<<END
  <table>
  <tr>
    <th>Department<br>ID</th>
    <th>Department<br>Name</th>
    <th>Manager<br>Id</th>
    <th>Location ID</th>
  </tr>
  <tr>
END;
    echo '<td>'.htmlentities($dept['DEPARTMENT_ID']).'</td>';
    echo '<td>'.htmlentities($dept['DEPARTMENT_NAME']).'</td>';
    echo '<td>'.htmlentities($dept['MANAGER_ID']).'</td>';
    echo '<td>'.htmlentities($dept['LOCATION_ID']).'</td>';
    echo <<<END
  </tr>
  </table>
END;
```

```
    }
  }
```

Remember the `END;` line must not be prefixed with leading spaces otherwise the rest of the document is treated as part of the text to be printed.

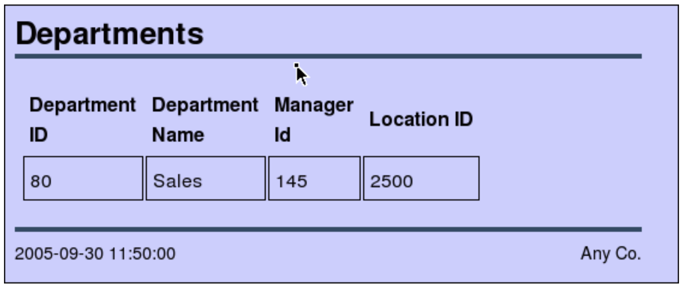
5. Edit `anyco.php`. Include `anyco_ui.inc` and `anyco_db.inc`, and call the database functions to query and display information for a department with a `department_id` of 80 by using the following code. The file becomes:

```
<?php // File: anyco.php

require('anyco_cn.inc');
require('anyco_db.inc');
require('anyco_ui.inc');

$query =
  'SELECT   department_id, department_name, manager_id, location_id
   FROM     departments
   WHERE    department_id = 80';

$conn = db_connect();

$dept = db_do_query($conn, $query);
ui_print_header('Departments');
ui_print_department($dept[0]);
ui_print_footer(date('Y-m-d H:i:s'));

?>
```

6. To test the resulting changes to the application, in a browser window enter the following URL:

```
http://localhost/~<username>/chap4/anyco.php
```

The page returned in the browser window should resemble the following page:



Departments

Department ID	Department Name	Manager Id	Location ID
80	Sales	145	2500

2005-09-30 11:50:00 Any Co.

Writing Queries with Bind Variables

Using queries with hard coded values in the WHERE clause may be useful for some situations. However, if the query conditional values need to change it is not appropriate to encode a value into the query. Oracle recommends you use bind variables in the query as a placeholder replacing literal values in the query conditions.

A bind variable is a symbolic name preceded by a colon in the query that acts as a placeholder for literal values in the WHERE clause. For example, the query string created in the anyco.php file could be rewritten with the bind variable ":did":

```
$query =
  'SELECT   department_id, department_name, manager_id, location_id
```

```
FROM     departments
WHERE    department_id = :did';
```

By using bind variables to parameterize SQL statements:

- The statement is reusable with different input values without needing to change the code.
- The query performance is improved through a reduction of the query parse time in the server, since the Oracle database can reuse parse information from the previous invocations of the identical query string.
- There is protection against "SQL Injection" security problems.
- There is no need to specially handle quotes in user input.

When a query uses a bind variable the PHP code must associate an actual value with each bind variable (placeholder) used in the query before it is execute. This process is known as run-time binding.

To enable you PHP application to use bind variables in the query perform the following changes to your PHP application code:

1. Edit anyco.php. Modify the query to use a bind variable, create an array to store the value to be associated with the bind variable, and pass `$bindargs` to `db_do_query()`:

   ```
   <?php // File: anyco.php
   ...

   $query =
   'SELECT   department_id, department_name, manager_id, location_id
    FROM     departments
    WHERE    department_id = :did';

   $bindargs = array();
   // In the $bindargs array add an array containing
   // the bind variable name used in the query, its value, a length
   array_push($bindargs, array('DID', 80, -1));

   $conn = db_connect();
   $dept = db_do_query($conn, $query, $bindargs);

   ...
   ?>
   ```

   In this example, the bind variable, called DID, is an input argument in the parameterized query, and it is associated with the value 80. Later the value of the bind variable will be dynamically determined. In addition, the length component is passed as -1 as the OCI8 layer can determine the length. This is not the case for bind variables accepting output results from a query.

2. Edit `anyco_db.inc`. Modify the `db_do_query()` function to accept a `$bindvars` array variable as a third parameter. Call the `oci_bind_by_name()` OCI8 call to associate the PHP values supplied in `$bindvars` parameter with bind variables in the query:

   ```
   <?php // File: anyco_db.inc
   ...

   function db_do_query($conn, $statement, $bindvars = array())
   {
     $stid = oci_parse($conn, $statement);
   ```

```
    if (!$stid) {
      db_error($conn, __FILE__, __LINE__);
    }

    // Bind the PHP values to query bind parameters
    foreach ($bindvars as $b) {
      // create local variable with caller specified bind value
      $$b[0] = $b[1];
      // oci_bind_by_name(resource, bv_name, php_variable, length)
      $r = oci_bind_by_name($stid, ":$b[0]", $$b[0], $b[2]);
      if (!$r) {
        db_error($stid, __FILE__, __LINE__);
      }
    }
    $r = oci_execute($stid, OCI_DEFAULT);

    ...
  }
  ...
  ?>
```

The binding is performed in the `foreach` loop before the `oci_execute()` is done.

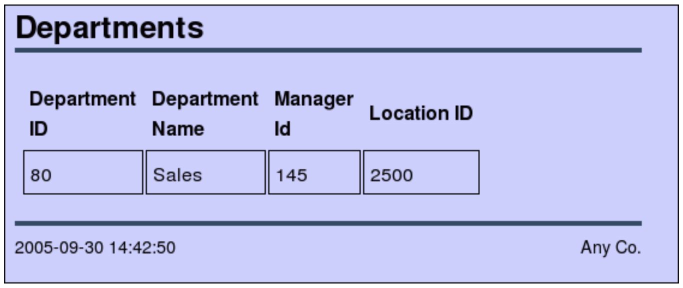
For each entry in `$bindvars` array, the first element contains the query bind variable name that is used to create a PHP variable of the same name, that is, `$$b[0]` takes the value 'DID' in `$b[0]` and forms a PHP variable called `$DID` whose value is assigned from the second element in the entry.

The `oci_bind_by_name()` accepts four parameters: the `$stid` as the resource, a string representing the bind variable name in the query derived from the first element in the array entry, the PHP variable containing the value to be associated with the bind variable, and the length of the input value.

3. To test the results of the preceding modifications, save the anyco.php and anyco_db.inc files and enter the following URL:

   `http://localhost/~<username>/chap4/anyco.php`

   The page returned in the browser window should resemble the following page:



Departments

Department ID	Department Name	Manager Id	Location ID
80	Sales	145	2500

2005-09-30 14:42:50 Any Co.

Navigating Through Database Records

Adding navigation through the database requires several important changes to the application logic. The modifications require the combination of:

- Including a HTML form to provide **Next** and **Previous** navigation buttons to step through data records.

- Detecting if the HTTP request for the page was posted by clicking the next or previous button.
- Tracking the last row queried by using HTTP session state. A PHP session is started to maintain state information for a specific client between HTTP requests. The first HTTP request will retrieve the first data row and initialize the session state. A subsequent request initiated with navigation buttons combined with the session state from a previous HTTP request enables the application to set variables that control the next record retrieved by the query.
- Writing a query that returns a sub set of rows based on a set of conditions whose values are determined by the application state.

To add navigation through database rows, perform the following steps:

1. Edit `anyco_ui.inc`. Add **Next** and **Previous** navigation buttons to the departments Web page. Change the `ui_print_departments()` function to append a second parameter called `$posturl` that supplies the value for the form attribute `action`. After printing the `</table>` tag include HTML form tags for the **Next** and **Previous** buttons:

```
<?php // File: anyco_ui.inc
...
function ui_print_department($dept, $posturl)
{
  ...
    echo <<<END
  </tr>
  </table>
  <form method="post" action="$posturl">
  <input type="submit" value="< Previous" name="prevdept">
  <input type="submit" value="Next >"     name="nextdept">
  </form>
END;
  }
}

?>
```

2. Edit `anyco.php`. To detect if the **Next** or **Previous** button was used to invoke the page and track session state, call the PHP function `session_start()`, and create a function named `construct_departments()`:

   Move and modify the database access logic into a new `construct_departments()` function, which detects if navigation has been performed, manages session state, defines a sub query for the database access layer to process, connects and calls a function `db_get_page_data()`. The file becomes:

```
<php // File: anyco.php

require('anyco_cn.inc');
require('anyco_db.inc');
require('anyco_ui.inc');

session_start();
construct_departments();

function construct_departments()
{
  if (isset($_SESSION['currentdept']) &&
      isset($_POST['prevdept']) &&
      $_SESSION['currentdept'] > 1) {
    $current = $_SESSION['currentdept'] - 1;
  }
  elseif (isset($_SESSION['currentdept']) &&
          isset($_POST['nextdept'])) {
    $current = $_SESSION['currentdept'] + 1;
  }
  elseif (isset($_POST['showdept']) &&
          isset($_SESSION['currentdept'])) {
    $current = $_SESSION['currentdept'];
  }
  else {
    $current = 1;
  }

  $query = 'SELECT department_id, department_name,
                   manager_id, location_id
            FROM   departments
            ORDER BY department_id asc';

  $conn = db_connect();

  $dept = db_get_page_data($conn, $query, $current, 1);
  $deptid = $dept[0]['DEPARTMENT_ID'];

  $_SESSION['currentdept'] = $current;

  ui_print_header('Department');
  ui_print_department($dept[0], $_SERVER['SCRIPT_NAME']);
  ui_print_footer(date('Y-m-d H:i:s'));
}

?>
```

The `if` and `elseif` construct at the start of the `construct_departments()` function is used to detect if a navigation button was with a HTTP post request to process the page, and tracks if the `currentdept` number is set in the session state. Depending on the circumstances, the variable `$current` is decremented by one when the previous button is clicked, `$current` is incremented by one when the **Next** button is clicked, otherwise `$current` is set to the current department, or initialized to one for the first time through.

A query is formed to obtain all the department rows in ascending sequence of the `department_id`. The `ORDER BY` clause is an essential part of the navigation logic. The query is used as a sub query inside the `db_get_page_data()` function to obtain a page of a number of rows, where the number of rows per page is specified as the fourth argument to the `db_get_page_data()` function. After connecting to the database, `db_get_page_data()` is called retrieve the set of rows obtained for the specified query. The `db_get_page_data()` function is provide with the connection resource, the query string, a value in `$current` specifying the first row in the next page of data rows required, and the number of rows per page (in this case one row per page).

After calling `db_get_page_data()` to obtain a page of rows, the value of `$current` is stored in the application session state.

Between printing the page header and footer, the `ui_print_department()` function is called to display the recently fetched department row, and uses `$_SERVER['SCRIPT_NAME']` to supply the current PHP script name for the

`$posturl` parameter to set the HTML form's action attribute. So each **Next** or **Previous** button click calls `anyco.php`.

3. Edit `anyco_db.inc`. Implement the `db_get_page_data()` function to query a sub set of rows:

```
// Return subset of records
function db_get_page_data($conn, $q1, $current = 1,
                 $rowsperpage = 1, $bindvars = array())
{
  // This query wraps the supplied query, and is used
  // to retrieve a subset of rows from $q1
  $query = 'SELECT *
            FROM (SELECT A.*, ROWNUM AS RNUM
                  FROM ('.$q1.') A
                  WHERE ROWNUM <= :LAST)
            WHERE :FIRST <= RNUM';

  // Set up bind variables.
  array_push($bindvars, array('FIRST', $current, -1));
  array_push($bindvars,
             array('LAST', $current+$rowsperpage-1, -1));

  $r = db_do_query($conn, $query, $bindvars);
  return($r);
}
```

The structure of the query in `db_get_page_data()` enables navigation through a set (or page) of database rows.

The query supplied in `$q1` is nested as a sub query inside the sub query

```
SELECT A.*, ROWNUM AS RNUM FROM $q1 WHERE ROWNUM <= :LAST
```

Remember the query supplied in `$q1` retrieves an ordered set of rows, which is filtered by its enclosing query to return all the rows from the first row to the next page size (`$rowsperpage`) of rows. This is possible since the Oracle `ROWNUM` function (or pseudo column) returns an integer number starting at 1 for each row returned by the query in `$q1`.

The set of rows, returned by the sub query enclosing query `$q1`, is filtered a second time by the condition in the outermost query

```
WHERE :FIRST <= RNUM
```

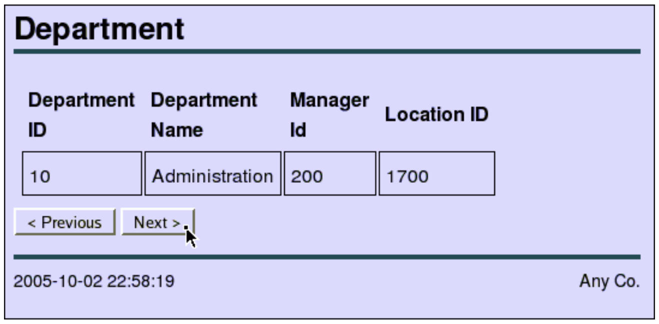
This condition ensures that rows prior to the value in `:FIRST` (the value in `$current`) are excluded from the final set of rows. The query enables navigation through a set rows where the first row is determined by the `$current` value and the page size is determined by the `$rowsperpage` value.

The `$current` value associated to the bind variable called `:FIRST`, the expression `$current+$rowsperpage-1` sets the value associated with the `:LAST` bind variable.

4. To test the changes made to your application, save the each file you modified, and enter the following URL in your Web browser:

```
http://localhost/~<username>/chap4/anyco.php
```

Since this is the first time you request the `anyco.php` page, you see the Administration department displayed:



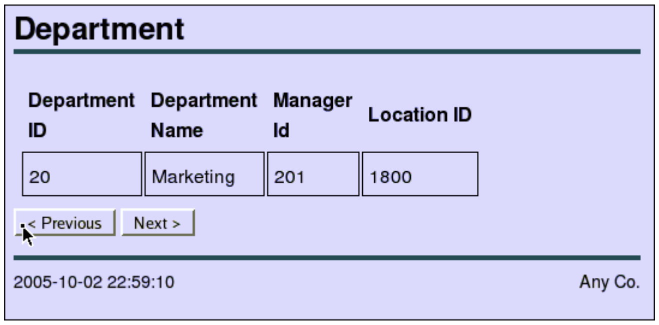
Department

Department ID	Department Name	Manager Id	Location ID
10	Administration	200	1700

< Previous Next >

2005-10-02 22:58:19 Any Co.

5. To navigate to the next department record (Marketing), click **Next**:



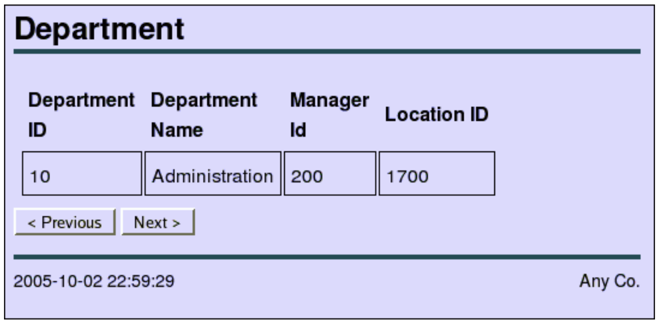
Department

Department ID	Department Name	Manager Id	Location ID
20	Marketing	201	1800

< Previous Next >

2005-10-02 22:59:10 Any Co.

6. To navigate back to the first department record (Administration), click **Previous**:



Department

Department ID	Department Name	Manager Id	Location ID
10	Administration	200	1700

< Previous Next >

2005-10-02 22:59:29 Any Co.

You may continue to test and experiment with the application by clicking **Next** and **Previous** to navigate to other records in the DEPARTMENTS table, as desired.

Note: If you navigate past the last record in the DEPARTMENTS table, an error will occur. Error handling is added in Adding Error Recovery in Chapter 5.

Extending the Basic Departments Form

The department tabular form is extended to include the following additional information:

- The department's manager name
- The number of employees assigned to the department
- The country name identifying the location of the department

The additional information is obtained by modifying the query to performing a join operation between the `DEPARTMENTS`, `EMPLOYEES`, `LOCATIONS`, and `COUNTRIES` tables.

To extend the department form, perform the following tasks:

1. Edit `anyco_ui.inc`. Modify the `ui_print_departments()` function by replacing the Manager ID and Location ID references with the Manager Name and Location, respectively, and insert a Number of Employees field after Department Name. Make the necessary changes in the table header and data fields. The function becomes:

```
function ui_print_department($dept, $posturl)
{
  if (!$dept) {
    echo '<p>No Department found</p>';
  }
  else {
    echo <<<END
  <table>
  <tr>
    <th>Department<br>ID</th>
    <th>Department<br>Name</th>
    <th>Number of<br>Employees</th>
    <th>Manager<br>Name</th>
    <th>Location</th>
  </tr>
  <tr>
END;
  echo '<td>'.htmlentities($dept['DEPARTMENT_ID']).'</td>';
  echo '<td>'.htmlentities($dept['DEPARTMENT_NAME']).'</td>';
  echo '<td>'.htmlentities($dept['NUMBER_OF_EMPLOYEES']).'</td>';
  echo '<td>'.htmlentities($dept['MANAGER_NAME']).'</td>';
  echo '<td>'.htmlentities($dept['COUNTRY_NAME']).'</td>';
  echo <<<END
  </tr>
  </table>
  <form method="post" action="$posturl">
  <input type="submit" value="< Previous" name="prevdept">
  <input type="submit" value="Next >"     name="nextdept">
  </form>
END;
  }
}
```

   There is no need to pass a `$bindargs` parameter to the `db_do_query()` call because we are not using bind variables. The `db_do_query()` declaration will provide a default value of an empty array automatically. PHP allows functions to have variable numbers of parameters.

2. Edit `anyco.php`. Replace the query string in `construct_departments()` with:

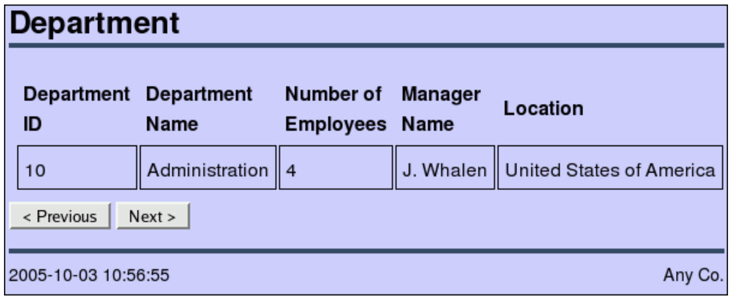
```
$query =
  "SELECT d.department_id, d.department_name,
       substr(e.first_name,1,1)||'. '|| e.last_name as manager_name,
       c.country_name, count(e2.employee_id) as number_of_employees
   FROM  departments d, employees e, locations l,
         countries c, employees e2
   WHERE d.manager_id = e.employee_id
   AND d.location_id = l.location_id
   AND d.department_id = e2.department_id
   AND l.country_id = c.country_id
   GROUP BY d.department_id, d.department_name,
         substr(e.first_name,1,1)||'. '||e.last_name,
         c.country_name
   ORDER BY d.department_id ASC";
```

The query string is enclosed in double quotes to simplify writing the statement which contains SQL literal strings in single quotes.

3. Save the changes to your files, and test the changes by entering the following URL in a Web browser:

   `http://localhost/~<username>/chap4/anyco.php`

   The Web page result should resemble the following output:



Department

Department ID	Department Name	Number of Employees	Manager Name	Location
10	Administration	4	J. Whalen	United States of America

Building the Basic Employee Form

To display employees, perform the following tasks:

1. Edit `anyco.php`. Add a function `construct_employees()` which constructs the employee query, calls `db_do_query()` to execute the query, and prints the results using `ui_print_employees()`:

```
function construct_employees()
{
  $query =
  "SELECT employee_id,
    substr(first_name,1,1) || '.  '|| last_name as employee_name,
    hire_date,
    to_char(salary, '9999G999D99') as salary,
    nvl(commission_pct,0) as commission_pct
   FROM   employees
   ORDER BY employee_id asc";

  $conn = db_connect();
  $emp = db_do_query($conn, $query);

  ui_print_header('Employees');
  ui_print_employees($emp);
  ui_print_footer(date('Y-m-d H:i:s'));
}
```

2. Edit `anyco.php`. Replace the call to `construct_departments()` with a call to `construct_employees()`:

```
<?php // File: anyco.php

require('anyco_cn.inc');
require('anyco_db.inc');
require('anyco_ui.inc');

session_start();
construct_employees();
...
?>
```

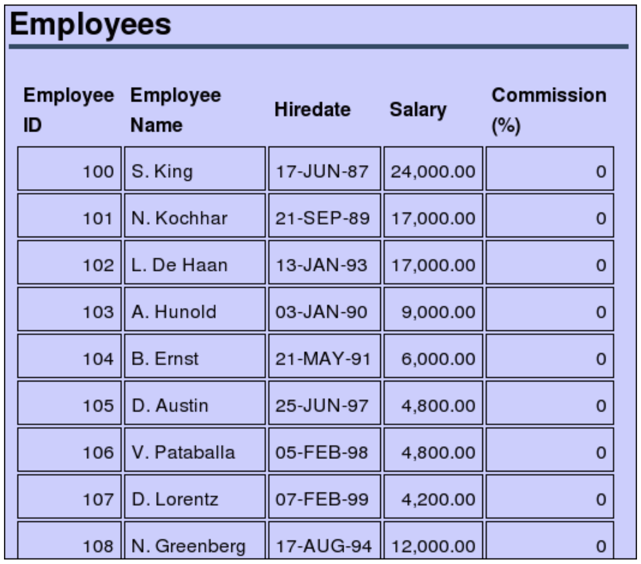
3. Edit `anyco_ui.inc`. Implement the presentation of employee data in a HTML table by adding a `ui_print_employees()` function:

```
function ui_print_employees($employeerecords)
{
  if (!$employeerecords) {
    echo '<p>No Employee found</p>';
  }
  else {
    echo <<<END
  <table>
  <tr>
    <th>Employee<br>ID</th>
    <th>Employee<br>Name</th>
    <th>Hiredate</th>
    <th>Salary</th>
    <th>Commission<br>(%)</th>
  </tr>
END;
    // Write one row per employee
    foreach ($employeerecords as $emp) {
      echo '<tr>';
      echo '<td align="right">'.
            htmlentities($emp['EMPLOYEE_ID']).'</td>';
      echo '<td>'.htmlentities($emp['EMPLOYEE_NAME']).'</td>';
      echo '<td>'.htmlentities($emp['HIRE_DATE']).'</td>';
      echo '<td align="right">'.
            htmlentities($emp['SALARY']).'</td>';
      echo '<td align="right">'.
            htmlentities($emp['COMMISSION_PCT']).'</td>';
      echo '</tr>';
    }
    echo <<<END
  </table>
END;
  }
}
```

4. Save the changes to `anyco.php` and `anyco_ui.inc`. Test the result of these changes by entering the following URL in your Web browser:

```
http://localhost/~<username>/chap4/anyco.php
```

Examine the result page, and scroll down to view all the employee rows displayed in the page:



Employees

Employee ID	Employee Name	Hiredate	Salary	Commission (%)
100	S. King	17-JUN-87	24,000.00	0
101	N. Kochhar	21-SEP-89	17,000.00	0
102	L. De Haan	13-JAN-93	17,000.00	0
103	A. Hunold	03-JAN-90	9,000.00	0
104	B. Ernst	21-MAY-91	6,000.00	0
105	D. Austin	25-JUN-97	4,800.00	0
106	V. Pataballa	05-FEB-98	4,800.00	0
107	D. Lorentz	07-FEB-99	4,200.00	0
108	N. Greenberg	17-AUG-94	12,000.00	0

5 Updating Data

In this chapter you extend the Anyco HR application with forms that enable you to insert, update, and delete an employee record.

Extending the Basic Employee Form

To enable employees records to manipulated, perform the following tasks:

1. Create the `chap5` directory and copy application files from `chap4`:

   ```
   mkdir $HOME/public_html/chap5
   cp $HOME/public_html/chap4/* $HOME/public_html/chap5
   cd $HOME/public_html/chap5
   ```

2. Edit `anyco.php`. Add form handler control logic to manage the requests for showing, inserting, updating, and deleting employees:

   ```
   <?php // File: anyco.php

   require('anyco_cn.inc');
   require('anyco_db.inc');
   require('anyco_ui.inc');

   session_start();
   // Start form handler code
   if (isset($_POST['insertemp'])) {
     construct_insert_emp();
   }
   elseif (isset($_POST['saveinsertemp'])) {
     insert_new_emp();
   }
   elseif (isset($_POST['modifyemp'])) {
     construct_modify_emp();
   }
   elseif (isset($_POST['savemodifiedemp'])) {
     modify_emp();
   }
   elseif (isset($_POST['deleteemp'])) {
     delete_emp();
   }
   ```

```
else {
  construct_employees();
}

...
```

3. Edit `anyco.php`. Add the `construct_insert_emp()` function:

```
function construct_insert_emp()
{
  $conn = db_connect();

  $query = "SELECT job_id, job_title
            FROM jobs
            ORDER BY job_title ASC";
  $jobs = db_do_query($conn, $query,
                      OCI_FETCHSTATEMENT_BY_COLUMN);

  $query = "SELECT sysdate FROM dual";
  $date = db_do_query($conn, $query,
                      OCI_FETCHSTATEMENT_BY_COLUMN);
  $emp = array(
    'DEPARTMENT_ID' => 10,      // Default to department 10
    'HIRE_DATE' => $date['SYSDATE'][0],
    'ALLJOBIDS' => $jobs['JOB_ID'],
    'ALLJOBTITLES' => $jobs['JOB_TITLE']
    );

  ui_print_header('Insert New Employee');
  ui_print_insert_employee($emp, $_SERVER['SCRIPT_NAME']);
  // Note: The two kinds of date used:
  // 1) SYSDATE for storing an SQL date in the database, and
  // 2) The PHP date for display in the footer of each page
  ui_print_footer(date('Y-m-d H:i:s'));
}
```

The `construct_insert_emp()` function executes two queries to obtain default data to be used to populate the insert employee form, which is displayed by the `ui_print_insert_employee()` function.

The `$query` of the `JOBS` table obtains a list of all the existing job ID's and their descriptions in order to build a list for selecting a job type in the HTML form generated by the `ui_print_insert_employee()` function.

The `$query` using `SYSDATE` obtains the current database date and time for setting the default hire date of the new employee.

There are two kinds of date used in the application code, the PHP `date()` function for printing the date and time in the page footer, and the Oracle `SYSDATE` function to obtain the default date and time for displaying in the employee HTML form's hire date field and to ensure the field text is entered in the correct database format.

The two `db_do_query()` function calls provide an additional parameter value `OCI_FETCHSTATEMENT_BY_COLUMNS` to specify that the return type for query is an array of column values.

4. Edit `anyco.php`. In the `construct_employees()` function modify the `db_do_query()` call to supply `OCI_FETCHSTATEMENT_BY_ROW` as the last parameter, and provide `$_SERVER['SCRIPT_NAME']` as second parameter in the `ui_print_employees()` call:

```
function construct_employees()
{
  $query =
  "SELECT employee_id,
    substr(first_name,1,1) || '.  '|| last_name as employee_name,
    hire_date,
    to_char(salary, '9999G999D99') as salary,
    nvl(commission_pct,0) as commission_pct
   FROM   employees
   ORDER BY employee_id asc";

  $conn = db_connect();
  $emp = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_ROW);

  ui_print_header('Employees');
  ui_print_employees($emp, $_SERVER['SCRIPT_NAME']);
  ui_print_footer(date('Y-m-d H:i:s'));
}
```

5. Edit `anyco.php`. Add `insert_new_emp()` to insert an employee into the `EMPLOYEES` table:

```
function insert_new_emp()
{
  $newemp = $_POST;
  $statement =
    "INSERT INTO employees
        (employee_id, first_name, last_name, email, hire_date,
         job_id, salary, commission_pct, department_id)
     VALUES (employees_seq.nextval, :fnm, :lnm, :eml, :hdt, :jid,
             :sal, :cpt, :did)";

  $conn = db_connect();
  $emailid = $newemp['firstname'].$newemp['lastname'];

  $bindargs = array();
  array_push($bindargs, array('FNM', $newemp['firstname'], -1));
  array_push($bindargs, array('LNM', $newemp['lastname'], -1));
  array_push($bindargs, array('EML', $emailid, -1));
  array_push($bindargs, array('HDT', $newemp['hiredate'], -1));
  array_push($bindargs, array('JID', $newemp['jobid'], -1));
  array_push($bindargs, array('SAL', $newemp['salary'], -1));
  array_push($bindargs, array('CPT', $newemp['commpct'], -1));
  array_push($bindargs, array('DID', $newemp['deptid'], -1));

  $r = db_execute_statement($conn, $statement, $bindargs);
  construct_employees();
}
```

The return value from `db_execute_statement()` is ignored and not even assigned to a variable, because we don't perform any action on its result until later.

6. Edit `anyco.php`. Add `construct_modify_emp()` to build the HTML form for updating an employee.

```
function construct_modify_emp()
{
  $empid = $_POST['emprec'];
  $query =
    "SELECT employee_id, first_name, last_name, email, hire_date,
            salary, nvl(commission_pct,0) as commission_pct
```

```
       FROM   employees
       WHERE  employee_id = :empid";

  $conn = db_connect();
  $bindargs = array();
  array_push($bindargs, array('EMPID', $empid, -1));

  $emp = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_ROW,
                                    $bindargs);

  ui_print_header('Modify Employee ');
  ui_print_modify_employee($emp[0], $_SERVER['SCRIPT_NAME']);
  ui_print_footer(date('Y-m-d H:i:s'));
}
```

7. Edit `anyco.php`. Add `modify_emp()` to update the employee row in the `EMPLOYEES` table, using the update form field values:

```
function modify_emp()
{
  $newemp = $_POST;
  $statement =
    "UPDATE employees
     SET   first_name = :fnm, last_name = :lnm, email = :eml,
           salary = :sal, commission_pct = :cpt
     WHERE employee_id = :eid";

  $conn = db_connect();
  $bindargs = array();
  array_push($bindargs, array('EID', $newemp['empid'], -1));
  array_push($bindargs, array('FNM', $newemp['firstname'], -1));
  array_push($bindargs, array('LNM', $newemp['lastname'], -1));
  array_push($bindargs, array('EML', $newemp['email'], -1));
  array_push($bindargs, array('SAL', $newemp['salary'], -1));
  array_push($bindargs, array('CPT', $newemp['commpct'], -1));

  $r = db_execute_statement($conn, $statement, $bindargs);
  construct_employees();
}
```

8. Edit `anyco.php`. Add `delete_emp()` to delete an employee row from the `EMPLOYEES` table:

```
function delete_emp()
{
  $empid = $_POST['emprec'];
  $statement = "DELETE FROM employees
                WHERE employee_id = :empid";

  $conn = db_connect();
  $bindargs = array();
  array_push($bindargs, array('EMPID', $empid, 10));
  $r = db_execute_statement($conn, $statement, $bindargs);

  construct_employees();
}
```

9. Edit `anyco_db.inc`. Add `$resulttype` as a third parameter to `db_do_query()`. Replace the last parameter value,

OCI_FETCHSTATEMENT_BY_ROW, in the oci_fetch_all() call with a variable so callers can choose the output type.

```
function db_do_query($conn, $statement, $resulttype,
                     $bindvars = array())
{
  $stid = oci_parse($conn, $statement);

  ...

  $r = oci_fetch_all($stid, $results, null, null, $resulttype);
  return($results);
}
```

10. Edit anyco_db.inc. Inside the db_get_page_data() function insert OCI_FETCHSTATEMENT_BY_ROW as the third parameter value in the db_do_query() call:

```
function db_get_page_data($conn, $q1, $current = 1,
                          $rowsperpage = 1, $bindvars = array())
{

  ...

  $r = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_ROW, $bindvars);
  return($r);
}
```

11. Edit anyco_db.inc. Add a db_execute_statement() function to execute data manipulation statements:

```
function db_execute_statement($conn, $statement, $bindvars = array())
{
  $stid = oci_parse($conn, $statement);
  if (!$stid) {
    db_error($conn, __FILE__, __LINE__);
  }

  // Bind parameters
  foreach ($bindvars as $b) {
    // create local variable with caller specified bind value
    $$b[0] = $b[1];
    $r = oci_bind_by_name($stid, ":$b[0]", $$b[0], $b[2]);
    if (!$r) {
      db_error($stid, __FILE__, __LINE__);
    }
  }

  $r = oci_execute($stid);
  if (!$r) {
    db_error($stid, __FILE__, __LINE__);
  }
  return($r);
}
```

12. Edit anyco_ui.inc. Change ui_print_employees() to produce a HTML form containing the employee rows. The function becomes:

```
function ui_print_employees($employeerecords, $posturl)
{
  if (!$employeerecords) {
    echo '<p>No Employee found</p>';
  }
  else {
    echo <<<END
  <form method="post" action="$posturl">
  <table>
  <tr>
    <th>&nbsp;</th>
    <th>Employee<br>ID</th>
    <th>Employee<br>Name</th>
    <th>Hiredate</th>
    <th>Salary</th>
    <th>Commission<br>(%)</th>
  </tr>
END;
    // Write one row per employee
    foreach ($employeerecords as $emp) {
      echo '<tr>';
      echo '<td><input type="radio" name="emprec" value="'.
        htmlentities($emp['EMPLOYEE_ID']).'"></td>';
      echo '<td align="right">'.
        htmlentities($emp['EMPLOYEE_ID']).'</td>';
      echo '<td>'.htmlentities($emp['EMPLOYEE_NAME']).'</td>';
      echo '<td>'.htmlentities($emp['HIRE_DATE']).'</td>';
      echo '<td align="right">'.
        htmlentities($emp['SALARY']).'</td>';
      echo '<td align="right">'.
        htmlentities($emp['COMMISSION_PCT']).'</td>';
      echo '</tr>';
    }
    echo <<<END
  </table>
  <input type="submit" value="Modify" name="modifyemp">
  <input type="submit" value="Delete" name="deleteemp">
  &nbsp;&nbsp;
  <input type="submit" value="Insert new employee"
         name="insertemp">
  </form>
END;
  }
}
```

The form prints a radio button in the first column of each row to enable you to select the record to be modified or deleted.

13. Edit `anyco_ui.inc`. Add `ui_print_insert_employee()` to generate the form to input new employee data:

```
function ui_print_insert_employee($emp, $posturl)
{
  if (!$emp) {
    echo "<p>No employee details found</p>";
  }
  else {
    $deptid = htmlentities($emp['DEPARTMENT_ID']);
    $hiredate = htmlentities($emp['HIRE_DATE']);

    echo <<<END
  <form method="post" action="$posturl">
  <table>
    <tr>
```

```
        <td>Department ID</td>
        <td><input type="text" name="deptid" value="$deptid"
                   size="20"></td>
      </tr>
      <tr>
        <td>First Name</td>
        <td><input type="text" name="firstname" size="20"></td>
      </tr>
      <tr>
        <td>Last Name</td>
        <td><input type="text" name="lastname" size="20"></td>
      </tr>
      <tr>
        <td>Hiredate</td>
        <td><input type="text" name="hiredate" value="$hiredate"
                   size="20"></td>
      </tr>
      <tr>
        <td>Job</td>
         <td><select name="jobid">
END;
      // Write the drop down list of jobs
      for ($i = 0; $i < count($emp['ALLJOBIDS']); $i++)
      {
        echo '<option
               label="'.htmlentities($emp['ALLJOBTITLES'][$i]).'"'.
             ' value="'.htmlentities($emp['ALLJOBIDS'][$i]).'">'.
             htmlentities($emp['ALLJOBTITLES'][$i]).'</option>';
      }
      echo <<<END
        </select>
        </td>
      </tr>
      <tr>
        <td>Salary</td>
        <td><input type="text" name="salary" value="1"
                   size="20"></td>
      </tr>
      <tr>
        <td>Commission (%)</td>
        <td><input type="text" name="commpct" value="0"
                   size="20"></td>
      </tr>
    </table>
      <input type="submit" value="Save" name="saveinsertemp">
      <input type="submit" value="Cancel" name="cancel">
    </form>
END;
  }
}
```

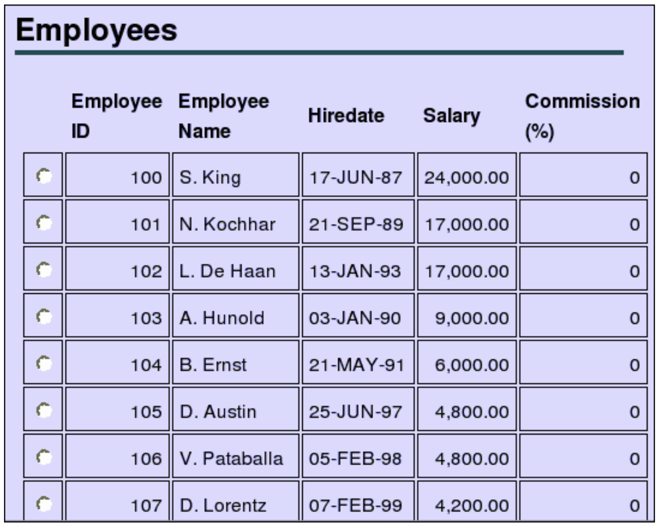
14. Edit `anyco_ui.inc`. Add `ui_print_modify_employee()` to generate the form to update an employee:

```
function ui_print_modify_employee($empdetails, $posturl)
{
  if (!$empdetails) {
    echo '<p>No Employee record selected</p>';
  }
  else {
    $fnm = htmlentities($empdetails['FIRST_NAME']);
    $lnm = htmlentities($empdetails['LAST_NAME']);
    $eml = htmlentities($empdetails['EMAIL']);
    $sal = htmlentities($empdetails['SALARY']);
    $cpt = htmlentities($empdetails['COMMISSION_PCT']);
    $eid = htmlentities($empdetails['EMPLOYEE_ID']);

    echo <<<END
  <form method="post" action="$posturl">
  <table>
    <tr>
      <td>Employee ID</td>
      <td>$eid</td></tr>
    <tr>
      <td>First Name</td>
      <td><input type="text" name="firstname" value="$fnm"></td>
    </tr>
    <tr>
      <td>Last Name</td>
      <td><input type="text" name="lastname" value="$lnm"></td>
    </tr>
    <tr>
      <td>Email Address</td>
      <td><input type="text" name="email" value="$eml"></td>
    </tr>
    <tr>
      <td>Salary</td>
      <td><input type="text" name="salary" value="$sal"></td>
    </tr>
    <tr>
      <td>Commission (%)</td>
      <td><input type="text" name="commpct" value="$cpt"></td>
    </tr>
  </table>
  <input type="hidden" value="{$empdetails['EMPLOYEE_ID']}"
         name="empid">
  <input type="submit" value="Save" name="savemodifiedemp">
  <input type="submit" value="Cancel" name="cancel">
  </form>
END;
  }
}
```

15. Save the changes to your Anyco application files, and test the changes by entering the following URL in you Web browser:

    `http://locahost/~<username>/chap5/anyco.php`

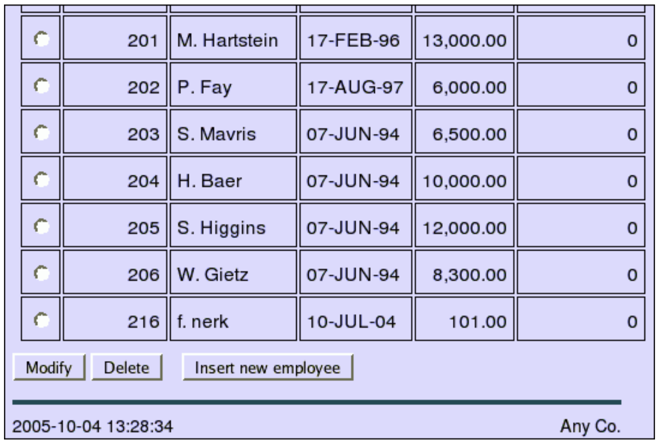
    The list of all employees is displayed with a radio button in each row.



Employees

	Employee ID	Employee Name	Hiredate	Salary	Commission (%)
○	100	S. King	17-JUN-87	24,000.00	0
○	101	N. Kochhar	21-SEP-89	17,000.00	0
○	102	L. De Haan	13-JAN-93	17,000.00	0
○	103	A. Hunold	03-JAN-90	9,000.00	0
○	104	B. Ernst	21-MAY-91	6,000.00	0
○	105	D. Austin	25-JUN-97	4,800.00	0
○	106	V. Pataballa	05-FEB-98	4,800.00	0
○	107	D. Lorentz	07-FEB-99	4,200.00	0

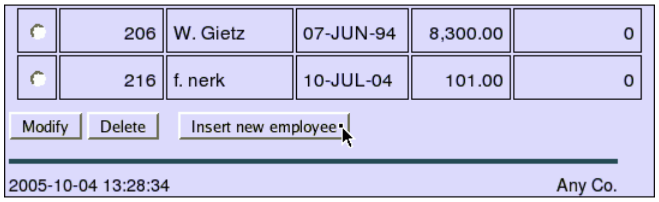
Scroll to the bottom of the Employees page to view the **Modify**, **Delete** and **Insert new employee** buttons:



○	201	M. Hartstein	17-FEB-96	13,000.00	0
○	202	P. Fay	17-AUG-97	6,000.00	0
○	203	S. Mavris	07-JUN-94	6,500.00	0
○	204	H. Baer	07-JUN-94	10,000.00	0
○	205	S. Higgins	07-JUN-94	12,000.00	0
○	206	W. Gietz	07-JUN-94	8,300.00	0
○	216	f. nerk	10-JUL-04	101.00	0

2005-10-04 13:28:34 Any Co.

16. To insert a new employee, click **Insert new employee**:



○	206	W. Gietz	07-JUN-94	8,300.00	0
○	216	f. nerk	10-JUL-04	101.00	0

2005-10-04 13:28:34 Any Co.

17. When you create or modify employees you will see that the database definitions require the salary to be greater than zero, and the commission to be less than 1. The commission will be rounded to two decimal places. In the Insert New Employee page, the Department ID field contains 10 (the default), Hiredate contains the current date (in default database date format), Salary contains 1, Commission (%) contains 0. Enter the following field values:

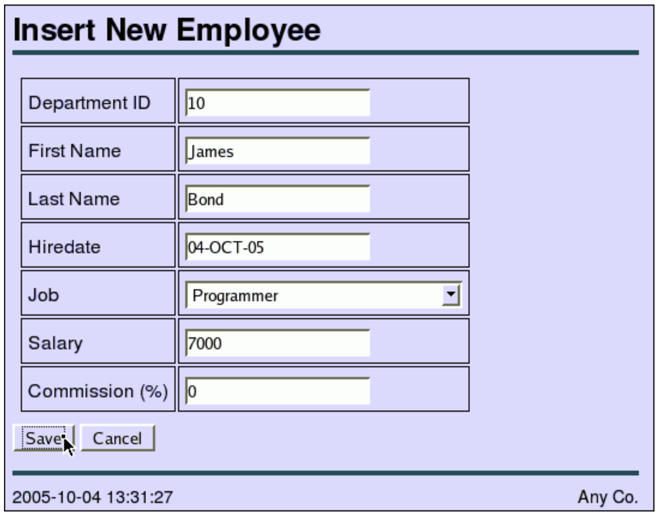
    First Name: James

    Last Name: Bond

    Job: Select Programmer from the drop down box.

    Salary: replace the 1 with 7000

Click **Save**.

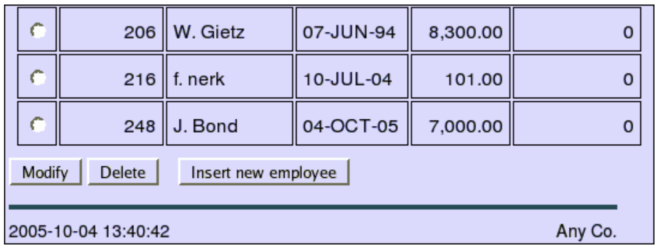


Insert New Employee

Department ID	10
First Name	James
Last Name	Bond
Hiredate	04-OCT-05
Job	Programmer
Salary	7000
Commission (%)	0

2005-10-04 13:31:27 Any Co.

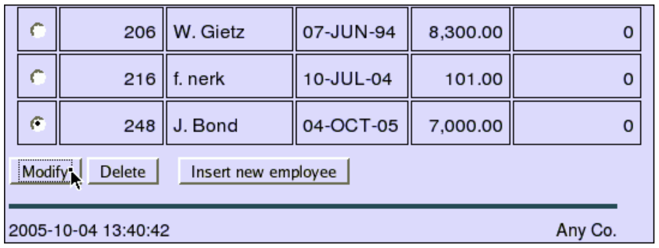
18. When the new employee is successfully inserted, the web page is refreshed with the form listing all employees. Scroll the web page to the last record and check that the new employee row is present. The employee ID assigned to the new record may be different on your system to the one shown in the following example:



○	206	W. Gietz	07-JUN-94	8,300.00	0
○	216	f. nerk	10-JUL-04	101.00	0
○	248	J. Bond	04-OCT-05	7,000.00	0

2005-10-04 13:40:42 Any Co.

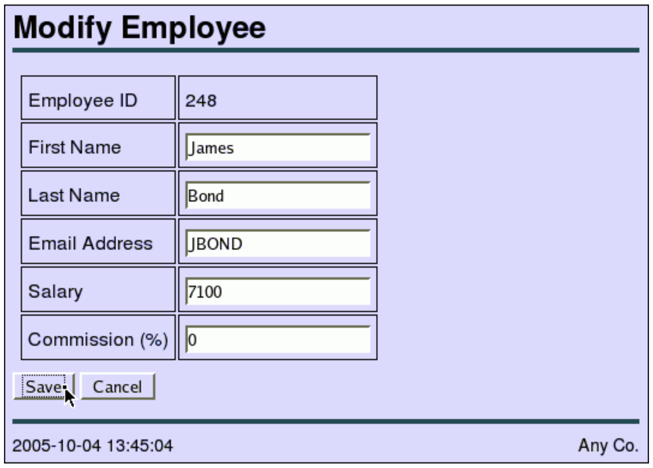
19. To modify the new employee, select the radio button next to the new employee row, click **Modify**:



○	206	W. Gietz	07-JUN-94	8,300.00	0
○	216	f. nerk	10-JUL-04	101.00	0
●	248	J. Bond	04-OCT-05	7,000.00	0

2005-10-04 13:40:42 Any Co.

20. In the Modify Employee page, modify the Email Address field to JBOND, and increase the Salary to 7100, click **Save**:

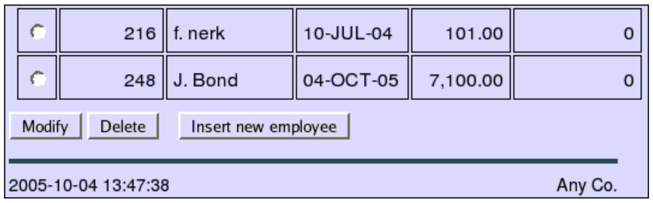


Modify Employee

Employee ID	248
First Name	James
Last Name	Bond
Email Address	JBOND
Salary	7100
Commission (%)	0

2005-10-04 13:45:04 Any Co.

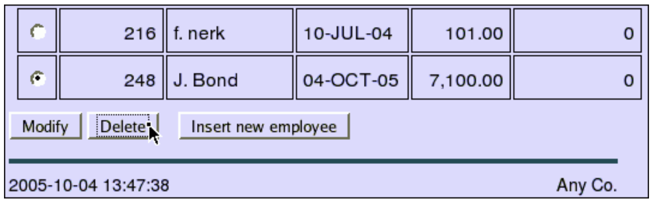
21. Successfully updating the employee causes the Employee page to be redisplayed. Scroll to the last employee row and confirm that the new employee's salary is now 7,100:



○	216	f. nerk	10-JUL-04	101.00	0
○	248	J. Bond	04-OCT-05	7,100.00	0

2005-10-04 13:47:38 Any Co.

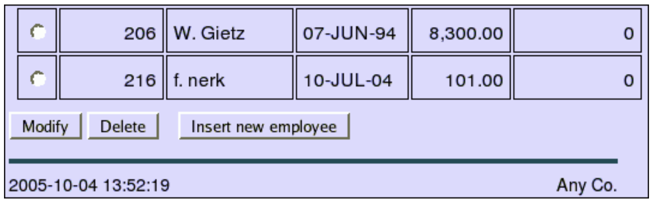
22. To remove the new employee row, select the radio button for the new employee row, click **Delete**:



○	216	f. nerk	10-JUL-04	101.00	0
●	248	J. Bond	04-OCT-05	7,100.00	0

2005-10-04 13:47:38 Any Co.

On successful deletion, the deleted row does not appear in the list of employees records redisplayed in the Employees page:



○	206	W. Gietz	07-JUN-94	8,300.00	0
○	216	f. nerk	10-JUL-04	101.00	0

2005-10-04 13:52:19 Any Co.

Combining Departments and Employees

1. Edit `anyco.php`. Modify the query in `construct_employees()` to include a `WHERE` clause to compare the `department_id` with a value in a bind variable called `:did`. This makes the page display employees in one department at a time. Get the `deptid` session parameter value to populate the bind variable:

```
$query =
 "SELECT employee_id,
         substr(first_name,1,1) || '.  '|| last_name as employee_name,
         hire_date,
         to_char(salary, '9999G999D99') as salary,
         nvl(commission_pct,0) as commission_pct
  FROM   employees
  WHERE  department_id = :did
  ORDER BY employee_id asc";

$deptid = $_SESSION['deptid'];
```

2. Edit `anyco.php`. In `construct_employees()`, update the call to `db_do_query()` to pass the bind information:

   ```
   $conn = oci_connect();

   $bindargs = array();
   array_push($bindargs, array('DID', $deptid, -1));

   $emp = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_ROW, $bindargs);
   ```

3. Edit `anyco.php`. In construct_departments() save the department identifier in a session parameter:

   ```
   $_SESSION['currentdept'] = $current;
   $_SESSION['deptid'] = $deptid;
   ```

   This saves the current department identifier from the Department page as a session parameter, which is used in the Employees page.

4. Edit `anyco.php`. Create a function `get_dept_name()` to query the department name for printing in the Department and Employee page titles:

   ```
   function get_dept_name($conn, $deptid)
   {
     $query =
       'SELECT department_name
        FROM   departments
        WHERE  department_id = :did';

     $conn = db_connect();
     $bindargs = array();
     array_push($bindargs, array('DID', $deptid, -1));
     $dn = db_do_query($conn, $query,OCI_FETCHSTATEMENT_BY_COLUMN, $bindargs);

     return($dn['DEPARTMENT_NAME'][0]);
   }
   ```

5. Edit `anyco.php`. Modify `construct_employees()` to print the department name in the page heading:

   ```
   $deptname = get_dept_name($conn, $deptid);
   ui_print_header('Employees: '.$deptname);
   ```

6. Edit `anyco.php`. Modify `construct_departments()` to print the department name in the page heading:

   ```
   $deptname = get_dept_name($conn, $deptid);
   ui_print_header('Department: '.$deptname);
   ```

7. Edit `anyco.php`. Modify `construct_insert_emp()` so the default department is obtained from the session parameter passed in the `$emp` array to `ui_print_insert_employee()`:

```
function construct_insert_emp()
{
  $deptid = $_SESSION['deptid'];

  $conn = db_connect();
  $query = "SELECT job_id, job_title FROM jobs ORDER BY job_title ASC";
  $jobs = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_COLUMN);
  $query = "SELECT sysdate FROM dual";
  $date = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_COLUMN);
  $emp = array(
    'DEPARTMENT_ID' => $deptid,
    'HIRE_DATE' => $date['SYSDATE'][0],
    'ALLJOBIDS' => $jobs['JOB_ID'],
    'ALLJOBTITLES' => $jobs['JOB_TITLE']
    );
  ui_print_header('Insert New Employee');
  ui_print_insert_employee($emp, $_SERVER['SCRIPT_NAME']);
  ui_print_footer(date('Y-m-d H:i:s'));
}
```

8. Edit `anyco.php`. Modify the final `else` statement in the HTML form handler. The handler becomes:

```
// Start form handler code
if (isset($_POST['insertemp'])) {
  construct_insert_emp();
}
elseif (isset($_POST['saveinsertemp'])) {
  insert_new_emp();
}
elseif (isset($_POST['modifyemp'])) {
  construct_modify_emp();
}
elseif (isset($_POST['savemodifiedemp'])) {
  modify_emp();
}
elseif (isset($_POST['deleteemp'])) {
  delete_emp();
}
elseif (   isset($_POST['showemp'])
        || isset($_POST['prevemp'])
        || isset($_POST['showemp'])) {
  construct_employees();
}
elseif (   isset($_POST['nextdept'])
        || isset($_POST['prevdept'])
        || isset($_POST['firstdept'])
        || isset($_POST['showdept'])) {
  construct_departments();
}
else {
  construct_departments();
}
```

9. Edit `anyco_ui.php`. In `ui_print_department()` change the HTML form to enable it to call the employee form:

```
...
<form method="post" action="$posturl">
<input type="submit" value="First" name="firstdept">
<input type="submit" value="< Previous" name="prevdept">
```

```
<input type="submit" value="Next >" name="nextdept">
&nbsp;&nbsp;&nbsp;
<input type="submit" value="Show Employees" name="showemp">
</form>
```

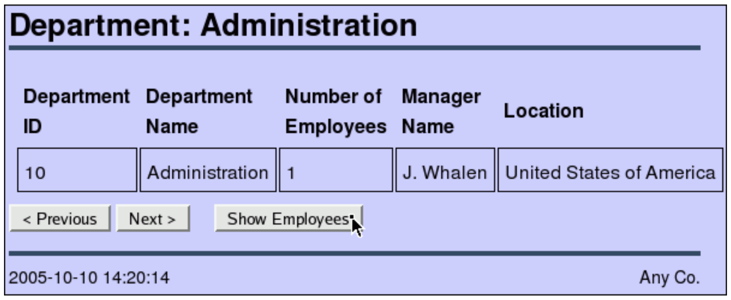
10. Edit `anyco_ui.php`. In `ui_print_employees()` change the HTML form to enable it to call the department form:

```
...
</table>
<input type="submit" value="Modify" name="modifyemp">
<input type="submit" value="Delete" name="deleteemp">
&nbsp;&nbsp;
<input type="submit" value="Insert new employee" name="insertemp">
&nbsp;&nbsp;
<input type="submit" value="Return to Departments" name="showdept">
</form>
```

11. Save the changes to your PHP files. In your browser, test the changes by entering the following URL:

    `http://localhost/~<username>/chap5/anyco.php`

    The Department Information page is displayed.

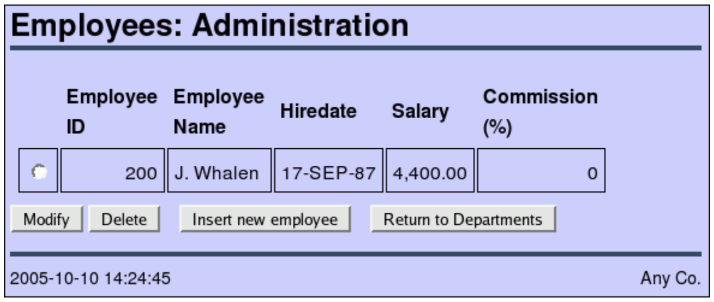


Department: Administration

Department ID	Department Name	Number of Employees	Manager Name	Location
10	Administration	1	J. Whalen	United States of America

2005-10-10 14:20:14 Any Co.

To display a list of employees in the department, click the **Show Employees** button.



Employees: Administration

	Employee ID	Employee Name	Hiredate	Salary	Commission (%)
○	200	J. Whalen	17-SEP-87	4,400.00	0

2005-10-10 14:24:45 Any Co.

You can return to the Department view by clicking the **Return to Departments** button. Experiment by navigating to another department and listing its employees to show the process of switching between the Department and Employee forms.

Adding Error Recovery

Error management is always a significant design decision. In production systems you might want to classify errors and handle them in different ways. Fatal errors could be redirected to a standard "site not available" page or home page. Data errors for new record creation might return to the appropriate form with invalid fields highlighted.

Most production systems would display_errors configuration option in the php.ini file set off, and log_errors set on.

PHP's output buffering functionality can be used to trap error text during a function. Using ob_start() prevents text from displaying on the screen. If an error occurs ob_get_contents() allows the previously generated error messages to be stored in a string for later display or analysis.

Here we'll change the application so error and database errors are displayed on a new page using a customer error handling function. Errors are now returned from the db* functions keeping them silent.

1. Edit anyco_db.inc. Change db_error() to return the error information in an array structure, instead of printing and quitting.:

```
function db_error($r = false, $file, $line)
{
  $err =  $r ? oci_error($r) : oci_error();

  if (isset($err['message'])) {
    $m = htmlentities($err['message']);
    $c = $err['code'];
  }
  else {
    $m = 'Unknown DB error';
    $c = null;
  }

  $rc = array(
    'MESSAGE' => $m,
    'CODE'    => $c,
    'FILE'    => $file,
    'LINE'    => $line
    );
  return $rc;
}
```

2. Edit anyco_db.inc. For every call to db_error(), assign the return value to a variable called $e and add a return false; statement after each call:

```
if (<error test>)
{
  $e = db_error(<handle>, __FILE__, __LINE__);
  return false;
}
```

   Make sure to keep the <error test> and <handle> parameters the same as they are currently specified for each call. Remember the __FILE__ and __LINE__ constants help pinpoint the location of the failure during development. This is useful information to log for fatal errors in a production deployment of an application.

3. Edit anyco_db.inc. Add a $e parameter to every function to enable the return of error information. Use the & reference prefix to ensure that results are returned to the calling function. Each function declaration becomes:

```
function db_connect(&$e) {...}

function db_get_page_data($conn, $q1, $currrownum = 1, $rowsperpage = 1,
                          &$e, $bindvars = array()) {...}

function db_do_query($conn, $statement, $resulttype, &$e,
                              $bindvars = array()) {...}

function db_execute_statement($conn, $statement, &$e,
                              $bindvars = array()) {...}
```

4. Edit `anyco_db.inc`. In `db_get_page_data()` change the call to `db_do_query()` to pass down the error parameter `$e`:

   ```
   $r = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_ROW, $e, $bindvars);
   ```

5. Edit `anyco.php`. Add an `@` prefix to all `oci_*` function calls.

   The `@` prefix prevents errors from displaying because each return result is tested. Preventing errors from displaying can hide incorrect parameter usage which may hinder testing the changes in this section.

6. Edit `anyco.php`. Create a function to handle the error information:

   ```
   function handle_error($message, $err)
   {
     ui_print_header($message);
     ui_print_error($err, $_SERVER['SCRIPT_NAME']);
     ui_print_footer(date('Y-m-d H:i:s'));
   }
   ```

7. Edit `anyco.php`. Modify all calls to `db_*` functions to include the additional error parameter:

   - Change all `db_connect()` calls to `db_connect($err)`.
   - Change all `db_do_query()` calls and insert a `$err` parameter as the fourth parameter.
   - Change the `db_get_page_data()` call and insert a `$err` parameter as the fifth parameter.
   - Change the `db_execute_statement()` calls and insert a `$err` parameter as the third parameter.

8. Edit `anyco.php`. Modify `construct_departments()` to handle errors returned. The function becomes:

   ```
   function construct_departments()
   {
     if (isset($_SESSION['currentdept']) && isset($_POST['prevdept']) &&
              $_SESSION['currentdept'] > 1)
        $current = $_SESSION['currentdept'] - 1;
     elseif (isset($_SESSION['currentdept']) && isset($_POST['nextdept']))
        $current = $_SESSION['currentdept'] + 1;
     elseif (isset($_POST['showdept']) && isset($_SESSION['currentdept']))
        $current = $_SESSION['currentdept'];
     else
        $current = 1;

     $query =
       "SELECT d.department_id, d.department_name,
               substr(e.first_name,1,1)||'. '|| e.last_name as manager_name,
               c.country_name, count(e2.employee_id) as number_of_employees
        FROM   departments d, employees e, locations l,
               countries c, employees e2
        WHERE  d.manager_id    = e.employee_id
        AND    d.location_id   = l.location_id
        AND    d.department_id = e2.department_id
   ```

```
     AND    l.country_id    = c.country_id
     GROUP BY d.department_id, d.department_name,
              substr(e.first_name,1,1)||'. '||e.last_name, c.country_name
     ORDER BY d.department_id ASC";

  $conn = db_connect($err);

  if (!$conn) {
    handle_error('Connection Error', $err);
  }
  else {
    $dept = db_get_page_data($conn, $query, $current, 1, $err);
    if ($dept === false) {
      // Use === so empty array at end of fetch is not matched
      handle_error('Cannot fetch Departments', $err);
    } else {

      if (!isset($dept[0]['DEPARTMENT_ID']) && $current > 1) {
        // no more records so go back one

        $current--;
        $dept = db_get_page_data($conn, $query, $current, 1, $err);
      }

      $deptid = $dept[0]['DEPARTMENT_ID'];

      $_SESSION['deptid'] = $deptid;
      $_SESSION['currentdept'] = $current;

      $deptname = get_dept_name($conn, $deptid);
      ui_print_header('Department: '.$deptname);
      ui_print_department($dept[0], $_SERVER['SCRIPT_NAME']);
      ui_print_footer(date('Y-m-d H:i:s'));
    }
  }
}
```

9. Edit `anyco.php`. Modify `construct_employees()` to handle errors. The function becomes:

```
function construct_employees()
{
  $query =
    "SELECT employee_id,
            substr(first_name,1,1) || '.  '|| last_name as employee_name,
            hire_date,
            to_char(salary, '9999G999D99') as salary,
            nvl(commission_pct,0) as commission_pct
     FROM   employees
     WHERE  department_id = :did
     ORDER BY employee_id asc";

  $deptid = $_SESSION['deptid'];

  $conn = db_connect($err);

  if (!$conn) {
    handle_error('Connection Error', $err);
  }
  else {
    $bindargs = array();
```

```
      array_push($bindargs, array('DID', $deptid, -1));
      $emp = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_ROW, $err,
      $bindargs);

      if (!$emp) {
        handle_error('Cannot fetch Employees', $err);
      }
      else {
        $deptname = get_dept_name($conn, $deptid);
        ui_print_header('Employees: '.$deptname);
        ui_print_employees($emp, $_SERVER['SCRIPT_NAME']);
        ui_print_footer(date('Y-m-d H:i:s'));
      }
    }
  }
```

10. Edit `anyco.php`. Modify `construct_insert_emp()` to handle errors. The function becomes:

```
function construct_insert_emp()
{
  $deptid = $_SESSION['deptid'];
  $conn = db_connect($err);
  if (!$conn) {
    handle_error('Connection Error', $err);
  }
  else {
    $query = "SELECT job_id, job_title FROM jobs ORDER BY job_title ASC";
    $jobs = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_COLUMN, $err);
    $query = "SELECT sysdate FROM dual";
    $date = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_COLUMN, $err);

    $emp = array(
      'DEPARTMENT_ID' => $deptid,
      'HIRE_DATE' => $date['SYSDATE'][0],
      'ALLJOBIDS' => $jobs['JOB_ID'],
      'ALLJOBTITLES' => $jobs['JOB_TITLE']
      );

    ui_print_header('Insert New Employee');
    ui_print_insert_employee($emp, $_SERVER['SCRIPT_NAME']);
    ui_print_footer(date('Y-m-d H:i:s'));
  }
}
```

11. Edit `anyco.php`. Modify `insert_new_emp()` to handle errors. The function becomes:

```
function insert_new_emp()
{
  $statement =
    'INSERT INTO employees
                 (employee_id, first_name, last_name, email, hire_date,
                 job_id, salary, commission_pct, department_id)
     VALUES (employees_seq.nextval, :fnm, :lnm, :eml, :hdt,
            :jid, :sal, :cpt, :did)';

  $newemp = $_POST;

  $conn = db_connect($err);
  if (!$conn) {
    handle_error('Connect Error', $err);
  }
  else {
    $emailid = $newemp['firstname'].$newemp['lastname'];

    $bindargs = array();
    array_push($bindargs, array('FNM', $newemp['firstname'], -1));
    array_push($bindargs, array('LNM', $newemp['lastname'], -1));
    array_push($bindargs, array('EML', $emailid, -1));
    array_push($bindargs, array('HDT', $newemp['hiredate'], -1));
    array_push($bindargs, array('JID', $newemp['jobid'], -1));
    array_push($bindargs, array('SAL', $newemp['salary'], -1));
    array_push($bindargs, array('CPT', $newemp['commpct'], -1));
    array_push($bindargs, array('DID', $newemp['deptid'], -1));

    $r = db_execute_statement($conn, $statement, $err, $bindargs);
    if ($r) {
      construct_employees();
    }
    else {
      handle_error('Cannot insert employee', $err);
    }
  }
}
```

12. Edit `anyco.php`. Modify `construct_modify_emp()` to handle errors. The function becomes:

```
function construct_modify_emp()
{
  if (!isset($_POST['emprec'])) { // User didn't select a record
    construct_employees();
  }
  else {
    $empid = $_POST['emprec'];

    $query =
      "SELECT employee_id, first_name, last_name, email, hire_date,
              salary, nvl(commission_pct,0) as commission_pct
       FROM   employees
       WHERE  employee_id = :empid";

    $conn = db_connect($err);
    if (!$conn) {
      handle_error('Connect Error', $err);
    }
    else {
      $bindargs = array();
      array_push($bindargs, array('EMPID', $empid, -1));

      $emp = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_ROW, $err,
             $bindargs);

      if (!$emp) {
        handle_error('Cannot find details for employee '.$empid, $err);
      }
      else {
        ui_print_header('Modify Employee ');
        ui_print_modify_employee($emp[0], $_SERVER['SCRIPT_NAME']);
        ui_print_footer(date('Y-m-d H:i:s'));
      }
```

```
        }
      }
    }
```

13. Edit `anyco.php`. Change `modify_emp()` to handle errors. The function becomes:

```
    function modify_emp()
    {
      $newemp = $_POST;

      $statement =
        "UPDATE employees
         SET    first_name = :fnm, last_name = :lnm, email = :eml,
                salary = :sal, commission_pct = :cpt
         WHERE  employee_id = :eid";

      $conn = db_connect($err);
      if (!$conn) {
        handle_error('Connect Error', $err);
      }
      else {
        $bindargs = array();
        array_push($bindargs, array('EID', $newemp['empid'], -1));
        array_push($bindargs, array('FNM', $newemp['firstname'], -1));
        array_push($bindargs, array('LNM', $newemp['lastname'], -1));
        array_push($bindargs, array('EML', $newemp['email'], -1));
        array_push($bindargs, array('SAL', $newemp['salary'], -1));
        array_push($bindargs, array('CPT', $newemp['commpct'], -1));

        $r = db_execute_statement($conn, $statement, $err, $bindargs);

        if (!$r) {
          handle_error('Cannot update employee '.$newemp['empid'], $err);
        }
        else {
          construct_employees();
        }
      }
    }
```

14. Edit `anyco.php`. Modify `delete_emp` to handle errors. The function becomes:

```
    function delete_emp()
    {
      if (!isset($_POST['emprec'])) { // User didn't select a record
        construct_employees();
      }
      else {
        $empid = $_POST['emprec'];

        $conn = db_connect($err);
        if (!$conn) {
          handle_error('Connection Error', $err);
        }
        else {
          $statement = "DELETE FROM employees WHERE employee_id = :empid";
          $bindargs = array();
          array_push($bindargs, array('EMPID', $empid, -1));
          $r = db_execute_statement($conn, $statement, $err, $bindargs);

          if (!$r) {
            handle_error("Error deleting employee $empid", $err);
          }
          else {
            construct_employees();
          }
        }
      }
    }
```

15. Edit `anyco.php`. Modify `get_dept_name()` to handle errors. The function becomes:

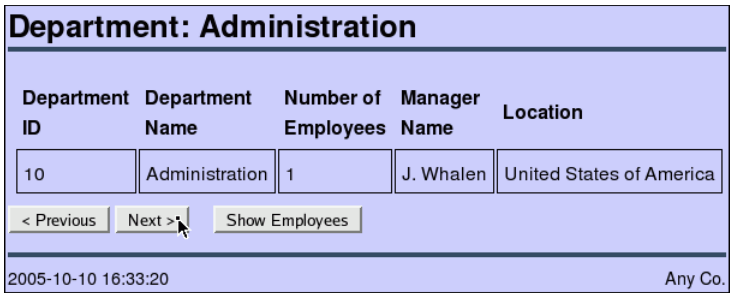
```
    function get_dept_name($conn, $deptid)
    {
      $query =
        'SELECT department_name
         FROM   departments
         WHERE  department_id = :did';

      $conn = db_connect($err);
      if (!$conn) {
        return ('Unknown');
      }
      else {
        $bindargs = array();
        array_push($bindargs, array('DID', $deptid, -1));
        $dn = db_do_query($conn, $query, OCI_FETCHSTATEMENT_BY_COLUMN,
                          $err, $bindargs);
        if ($dn == false)
          return ('Unknown');
        else
          return($dn['DEPARTMENT_NAME'][0]);
      }
    }
  }
```

16. Edit `afico_ui.inc`. Add a new function `ui_print_errors()`:

```
function ui_print_error($message, $posturl)
{
  if (!$message) {
    echo '<p>Unknown error</p>';
  }
  else {
    echo "<p>Error at line {$message['LINE']} of "
         ."{$message['FILE']}</p>";  // Uncomment for debugging
    echo "<p>{$message['MESSAGE']}</p>";
  }
  echo <<<END
  <form method="post" action="$posturl">
  <input type="submit" value="Return to Departments" name="showdept">
END;
}
```

17. Save the changes to your application files. Test the changes by entering the following URL in your browser:

```
http://localhost/~<username>/chap5/anyco.php
```

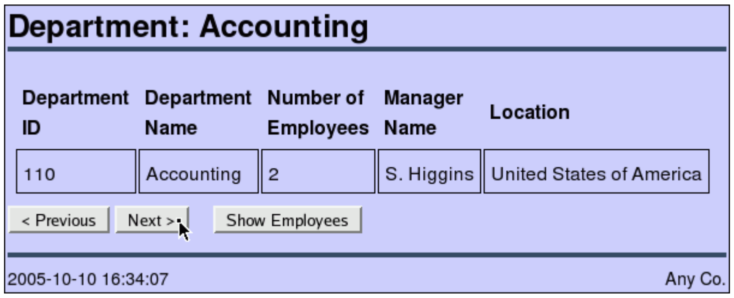


Department: Administration

Department ID	Department Name	Number of Employees	Manager Name	Location
10	Administration	1	J. Whalen	United States of America

2005-10-10 16:33:20 Any Co.

18. Click Next> to navigate to the last department record, the Accounting department with ID 110. Try to navigate past the last department record by clicking next.



Department: Accounting

Department ID	Department Name	Number of Employees	Manager Name	Location
110	Accounting	2	S. Higgins	United States of America

2005-10-10 16:34:07 Any Co.

The error handling prevents navigation past the last department record.

Further Error Handling

Specific Oracle errors can be handled individually. For example, if a new employee is created by clicking the **Insert new employee** button on the Employees page, and the Department ID is changed to a department that doesn't exist, we can trap this error and display a more meaningful message:

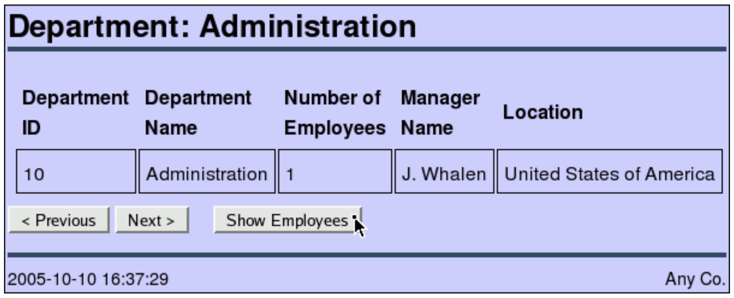
1. Edit `anyco.php`. Change the error handling in `insert_new_emp()`:

```
$r = db_execute_statement($conn, $statement, $err, $bindargs);
if ($r) {
  construct_employees();
}
else {
  if ($err['CODE'] == 2291) {  // Foreign key violated
    handle_error("Department {$newemp['deptid']} does not yet exist",
    $err);
  }
  else {
    handle_error('Cannot insert employee', $err);
  }
}
```

2. Save the changes to your application files. Test the changes by entering the following URL:

```
http://localhost/~<username>/chap5/anyco.php
```

3. In the Department page, click **Show Employees**.

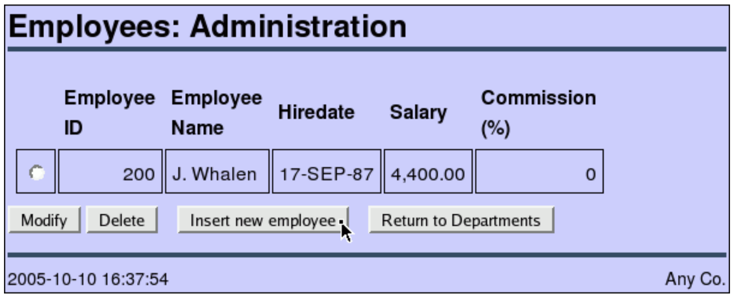


Department: Administration

Department ID	Department Name	Number of Employees	Manager Name	Location
10	Administration	1	J. Whalen	United States of America

2005-10-10 16:37:29 Any Co.

4. In the Employees page, click **Insert new employee**.

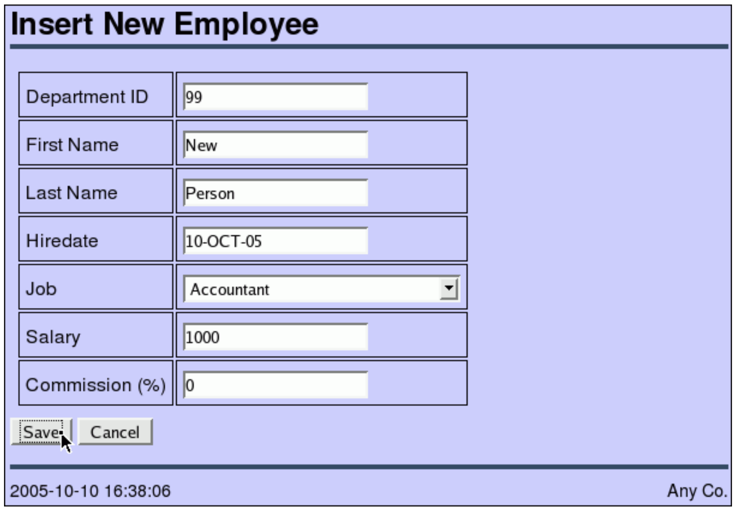


Employees: Administration

	Employee ID	Employee Name	Hiredate	Salary	Commission (%)
○	200	J. Whalen	17-SEP-87	4,400.00	0

2005-10-10 16:37:54 Any Co.

5. In the Insert New Employee page, enter employee details as shown setting the Department ID to 99. Click **Save**.

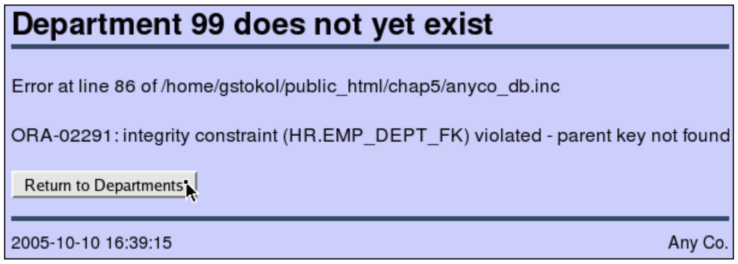


Insert New Employee

Department ID	99
First Name	New
Last Name	Person
Hiredate	10-OCT-05
Job	Accountant
Salary	1000
Commission (%)	0

2005-10-10 16:38:06 Any Co.

The following error page is displayed:



Department 99 does not yet exist

Error at line 86 of /home/gstokol/public_html/chap5/anyco_db.inc

ORA-02291: integrity constraint (HR.EMP_DEPT_FK) violated - parent key not found

2005-10-10 16:39:15 Any Co.

You can click **Return to Departments** to return to the Administration department page and then click **Show Employees** to verify that the new employee has not been added to the Administration department.

6

Executing Stored Procedures and Functions

This chapter shows you how to run stored procedures and functions using PHP and Oracle Express. It has the following topics:

- Using PL/SQL to Capture Business Logic
- Using PL/SQL Ref Cursors to Return Result Sets

The Anyco application is extended with a PL/SQL function to calculate remuneration for each employee, and further extended with a PL/SQL procedure to return a ref cursor of employee records.

Using PL/SQL to Capture Business Logic

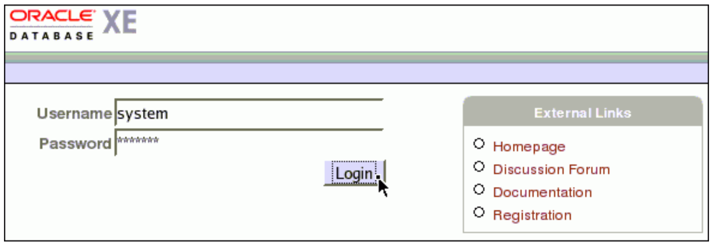
Oracle PL/SQL procedures and functions allow business logic to be stored in the database for any client program to use. They also reduce the amount of data that needs to be transferred between the database and PHP.

To display the total remuneration of each employee, perform the following steps to create a PL/SQL function stored in the database.

1. In a browser, enter the URL for your Oracle Database Express Edition HTMLDB page:

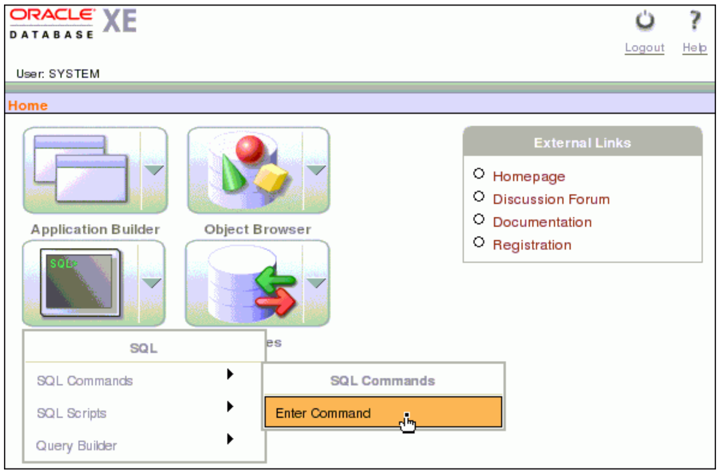
   ```
   http://localhost:8080/htmldb
   ```

2. At the login screen, in the Username field enter `system`, and in Password field enter `manager` (or the password you entered at the prompt during configuration of the Oracle Database Express Edition). Click **Login**.



Username: system
Password: *******

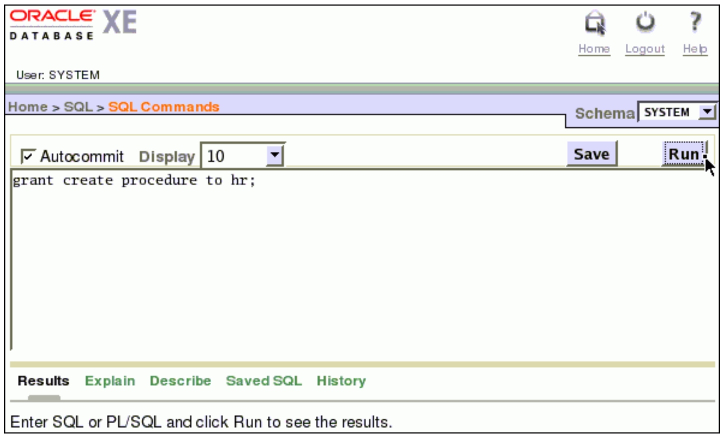
3. In the Home page, click the arrow on the **SQL** icon, move the mouse over **SQL Commands** and click **Enter Command**:



4. In the SQL Commands page, to assign the `create procedure` privilege to the `HR` user enter the following `grant` command:

```
grant create procedure to hr;
```

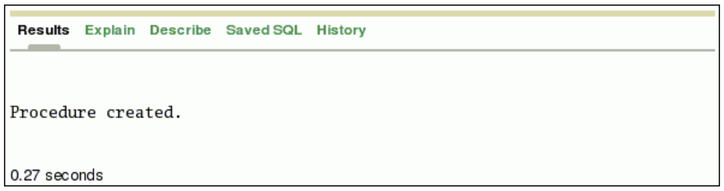
Click **Run**:



```
grant create procedure to hr;
```

Enter SQL or PL/SQL and click Run to see the results.

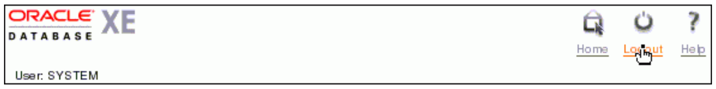
A message similar to the following appears in the Results section below the text area where the command was entered:



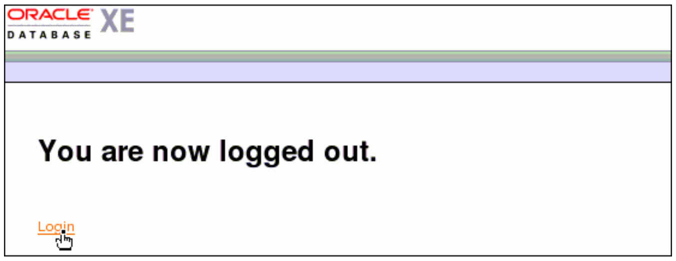
```
Procedure created.
```

0.27 seconds

5. Click the **Logout** link to terminate the HTMLDB session.

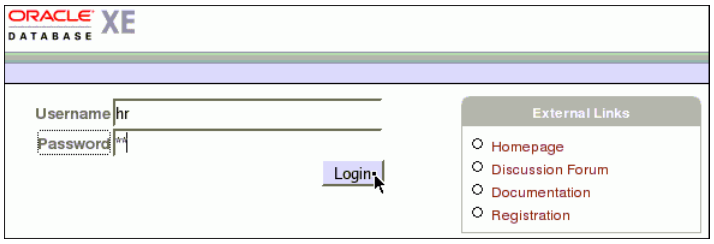


6. In the Logout Confirmation page, click the **Login** link:



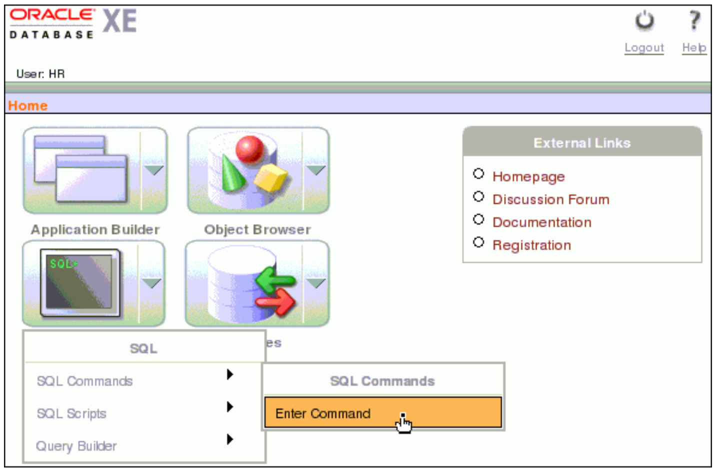
You are now logged out.

7. In the Oracle Database XE Login page, enter the Username `hr` and Password `hr`. Click **Login**:



Username hr
Password **

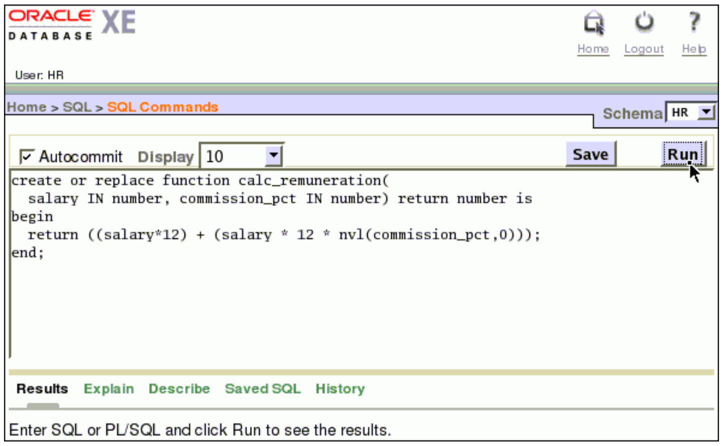
8. In the Home page, click the arrow on the **SQL** icon, move the mouse over **SQL Commands** and click **Enter Command**:



9. In the SQL Commands page, enter the following text to create a `calc_remuneration()` function:

```
create or replace function calc_remuneration(
  salary IN number, commission_pct IN number) return number is
begin
  return ((salary*12) + (salary * 12 * nvl(commission_pct,0)));
end;
```

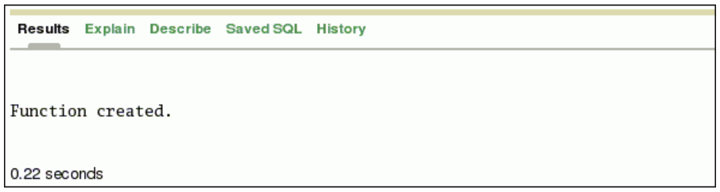
Click **Run**:



```
create or replace function calc_remuneration(
  salary IN number, commission_pct IN number) return number is
begin
  return ((salary*12) + (salary * 12 * nvl(commission_pct,0)));
end;
```

Enter SQL or PL/SQL and click Run to see the results.

In the results window, confirm that the function is created:



```
Function created.
```

0.22 seconds

10. Create the `chap6` directory and copy the application files from `chap5`:

```
mkdir $HOME/public_html/chap6
cp $HOME/public_html/chap5/* $HOME/public_html/chap6
cd $HOME/public_html/chap6
```

11. Edit `anyco.php`. Modify the query in `construct_employees()` to call the PL/SQL function for each row returned:

```
$query =
 "SELECT employee_id,
         substr(first_name,1,1) || '. '|| last_name as employee_name,
         hire_date,
         to_char(salary, '9999G999D99') as salary,
         nvl(commission_pct,0) as commission_pct,
         to_char(calc_remuneration(salary, commission_pct),'9999G999D99')
           as remuneration
  FROM employees
  WHERE department_id = :did
  ORDER BY employee_id ASC";
```

12. Edit `anyco_ui.inc`. In `ui_print_employees()` add a `Remuneration` column to the table, and modify the `foreach` loop to display the remuneration field for each employee:

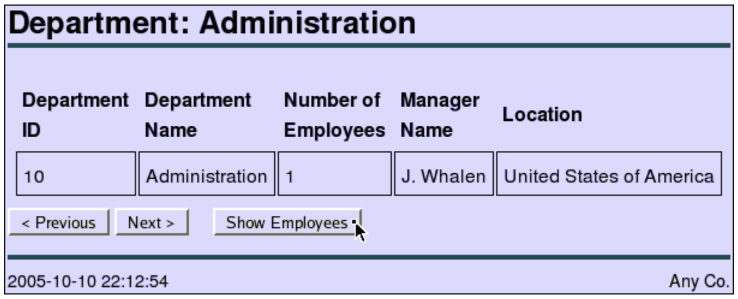
```
echo <<<END
   <form method="post" action="$posturl">
   <table>
   <tr>
     <th>&nbsp;</th>
     <th>Employee<br>ID</th>
     <th>Employee<br>Name</th>
     <th>Hiredate</th>
     <th>Salary</th>
     <th>Commission<br>(%)</th>
     <th>Remuneration</th>
```

```
      </tr>
   END;

       // Write one row per employee
       foreach ($employeerecords as $emp) {
         echo '<tr>';
         echo '<td><input type="radio" name="emprec"
                    value="'.htmlentities($emp['EMPLOYEE_ID']).'"></td>';
         echo '<td align="right">'.htmlentities($emp['EMPLOYEE_ID']).'</td>';
         echo '<td>'.htmlentities($emp['EMPLOYEE_NAME']).'</td>';
         echo '<td>'.htmlentities($emp['HIRE_DATE']).'</td>';
         echo '<td align="right">'.htmlentities($emp['SALARY']).'</td>';
         echo '<td align="right">'.htmlentities($emp['COMMISSION_PCT']).'</td>';
         echo '<td align="right">'.htmlentities($emp['REMUNERATION']).'</td>';
         echo '</tr>';
       }
```

13. Save the changes to your application files. In a browser, enter the following URL to test the application:

    `http://localhost/~<username>/chap6/anyco.php`

14. In the Departments form, click **Show Employees**.

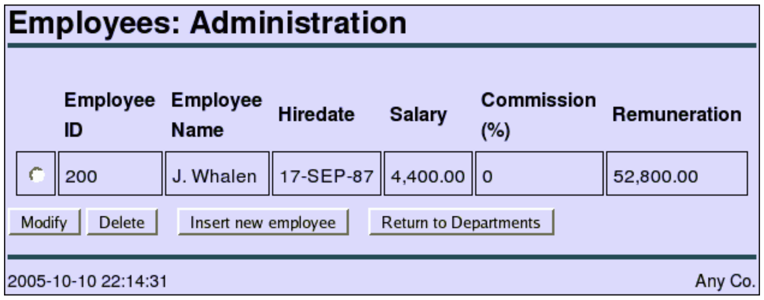


Department: Administration

Department ID	Department Name	Number of Employees	Manager Name	Location
10	Administration	1	J. Whalen	United States of America

2005-10-10 22:12:54 Any Co.

In the Employees page for the department, the employee remuneration is displayed in the last column:



Employees: Administration

	Employee ID	Employee Name	Hiredate	Salary	Commission (%)	Remuneration
○	200	J. Whalen	17-SEP-87	4,400.00	0	52,800.00

2005-10-10 22:14:31 Any Co.

Using PL/SQL Ref Cursors to Return Result Sets

Data sets can be returned as REF CURSORS from PL/SQL blocks in a PHP script. This can be useful where the dataset requires complex functionality.

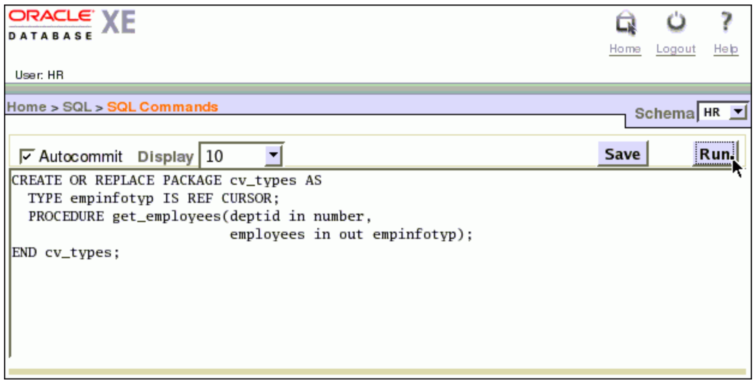
A REF CURSOR in PL/SQL is a type definition that is assigned to a cursor variable. It is common to declare a PL/SQL type inside a package specification for reuse in other PL/SQL constructs, such as a package body.

To create a PL/SQL package specification and body, with a REF CURSOR to retrieve employees for a specific department, perform the following steps:

1. In the SQL Commands page, as the HR user, create the following PL/SQL package specification:

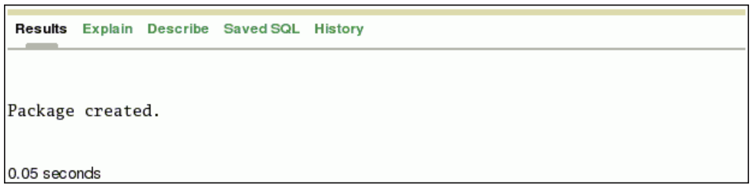
```
CREATE OR REPLACE PACKAGE cv_types AS
  TYPE empinfotyp IS REF CURSOR;
  PROCEDURE get_employees(deptid in number,
                          employees in out empinfotyp);
END cv_types;
```

Click **Run**:



```
CREATE OR REPLACE PACKAGE cv_types AS
  TYPE empinfotyp IS REF CURSOR;
  PROCEDURE get_employees(deptid in number,
                          employees in out empinfotyp);
END cv_types;
```

In the Results section, confirm the package specification is successfully created:



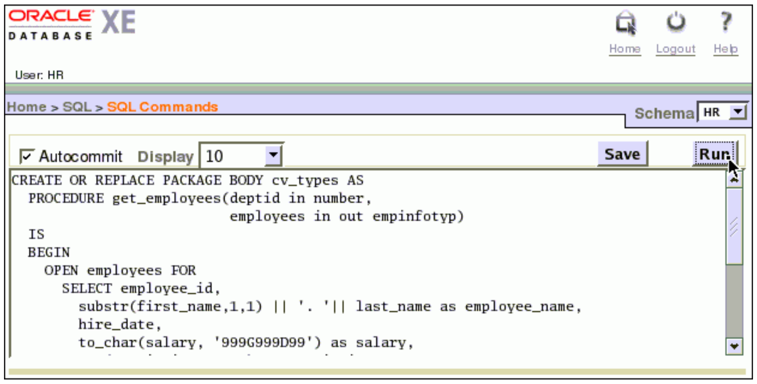
```
Package created.
```

0.05 seconds

2. In the SQL Commands page, as the HR user, create the PL/SQL package body (implementation):

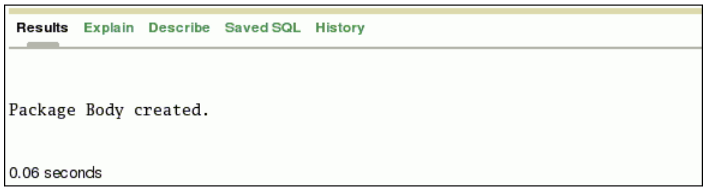
```
CREATE OR REPLACE PACKAGE BODY cv_types AS
  PROCEDURE get_employees(deptid in number,
                          employees in out empinfotyp)
  IS
  BEGIN
    OPEN employees FOR
      SELECT employee_id,
        substr(first_name,1,1) || '. '|| last_name as employee_name,
        hire_date,
        to_char(salary, '999G999D99') as salary,
        NVL(commission_pct,0) as commission_pct,
        to_char(calc_remuneration(salary, commission_pct),
                '9999G999D99') as remuneration
      FROM employees
      WHERE department_id = deptid
      ORDER BY employee_id ASC;
  END get_employees;
END cv_types;
```

Click **Run**:



```
CREATE OR REPLACE PACKAGE BODY cv_types AS
  PROCEDURE get_employees(deptid in number,
                          employees in out empinfotyp)
  IS
  BEGIN
    OPEN employees FOR
      SELECT employee_id,
        substr(first_name,1,1) || '. '|| last_name as employee_name,
        hire_date,
        to_char(salary, '999G999D99') as salary,
```

In the Results section, confirm the package body is successfully created:



```
Package Body created.
```

0.06 seconds

3. Edit `anyco_db.inc`. Create a new PHP function that calls the PL/SQL packaged procedure:

```
// Use ref cursor to fetch employee records
// All records are retrieved - there is no paging in this example
function db_get_employees_rc($conn, $deptid, &$e)
{
  // Excute the call to the stored procedure
  $stmt = "BEGIN cv_types.get_employees($deptid, :rc); END;";
  $stid = @oci_parse($conn, $stmt);
  if (!$stid) {
    $e = db_error($conn, __FILE__, __LINE__);
    return false;
  }
  $refcur = oci_new_cursor($conn);
  if (!$stid) {
    $e = db_error($conn, __FILE__, __LINE__);
    return false;
  }
  $r = @oci_bind_by_name($stid, ':RC', $refcur, -1, OCI_B_CURSOR);
  if (!$r) {
    $e = db_error($stid, __FILE__, __LINE__);
    return false;
  }
  $r = @oci_execute($stid);
  if (!$r) {
    $e = db_error($stid, __FILE__, __LINE__);
    return false;
  }
  // Now treat the ref cursor as a statement resource
  $r = @oci_execute($refcur, OCI_DEFAULT);
  if (!$r) {
    $e = db_error($refcur, __FILE__, __LINE__);
    return false;
  }
  $r = @oci_fetch_all($refcur, $employeerecords, null, null,
                      OCI_FETCHSTATEMENT_BY_ROW);
```

```
      if (!$r) {
        $e = db_error($refcur, __FILE__, __LINE__);
        return false;
      }
      return ($employeerecords);
    }
```

The `db_get_employees_rc()` function executes the following anonymous (unnamed) PL/SQL block:

```
BEGIN cv_types.get_employees($deptid, :rc); END;
```

The PL/SQL statement inside the "BEGIN END" block calls the stored PL/SQL package procedure `cv_types.et_employees()`. This returns an `OCI_B_CURSOR` ref cursor bind variable in the PHP variable `$refcur`.

The `$refcur` variable is treated as a statement handle that is used for execute and fetch operations.

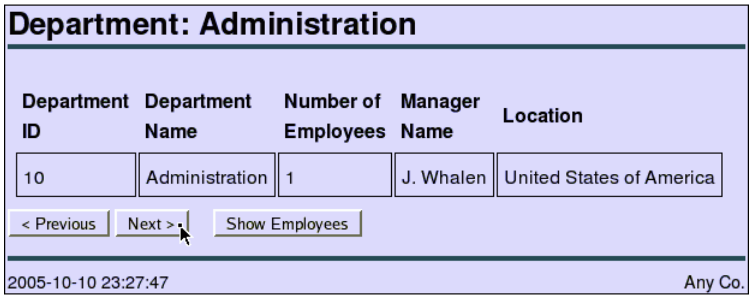
4. Edit `anyco.php`. Modify the `construct_employees()` function. Remove the query text and the bind arguments. The function becomes:

```
    function construct_employees()
    {
      $deptid = $_SESSION['deptid'];
      $conn = db_connect($err);
      if (!$conn) {
        handle_error('Connection Error', $err);
      }
      else {
        $emp = db_get_employees_rc($conn, $deptid, $err);

        if (!$emp) {
          handle_error('Cannot fetch Employees', $err);
        }
        else {
          $deptname = get_dept_name($conn, $deptid);

          ui_print_header('Employees: '.$deptname);
          ui_print_employees($emp, $_SERVER['SCRIPT_NAME']);
          ui_print_footer(date('Y-m-d H:i:s'));
        }
      }
    }
```

5. Save the changes to your application files. In a browser, enter the following URL to test the application:

```
http://localhost/~<username>/chap6/anyco.php
```

6. In the Departments form, click **Next >** to navigate to the Marketing department page.

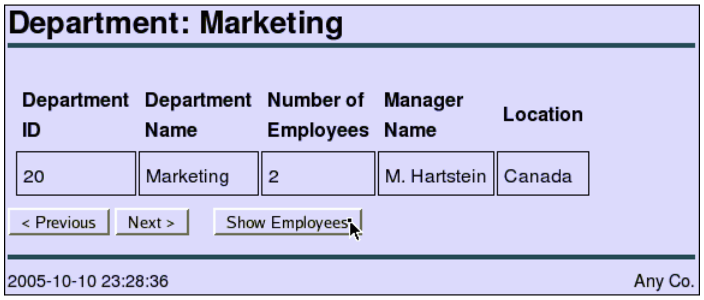


Department: Administration

Department ID	Department Name	Number of Employees	Manager Name	Location
10	Administration	1	J. Whalen	United States of America

2005-10-10 23:27:47 Any Co.

7. In the Marketing department page, click **Show Employees**.

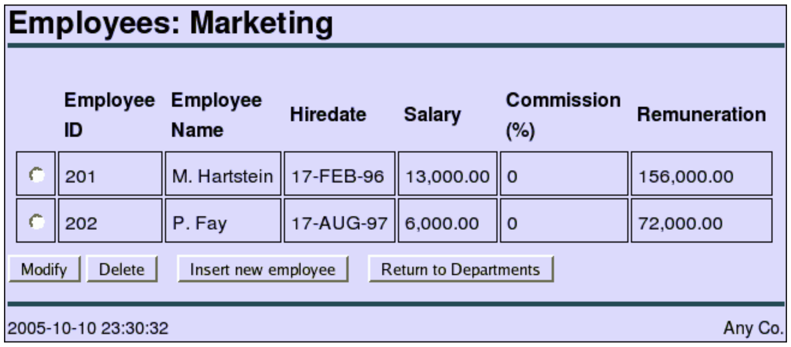


Department: Marketing

Department ID	Department Name	Number of Employees	Manager Name	Location
20	Marketing	2	M. Hartstein	Canada

2005-10-10 23:28:36 Any Co.

In the Employees page for the Marketing department, the employee records remuneration is displayed in the last column:

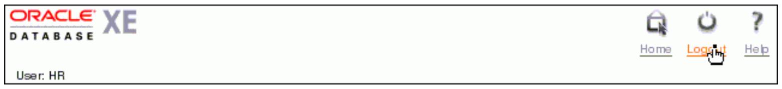


Employees: Marketing

	Employee ID	Employee Name	Hiredate	Salary	Commission (%)	Remuneration
○	201	M. Hartstein	17-FEB-96	13,000.00	0	156,000.00
○	202	P. Fay	17-AUG-97	6,000.00	0	72,000.00

2005-10-10 23:30:32 Any Co.

8. In the SQL Commands page, to log out of the HR database session, click the **Logout** link.



7 Loading Images

This chapter shows you how to change the application to upload a thumbnail picture for new employees and display it on the Employees page. It has the following topics:

Using Oracle LOBs to Store and Load Employee Images

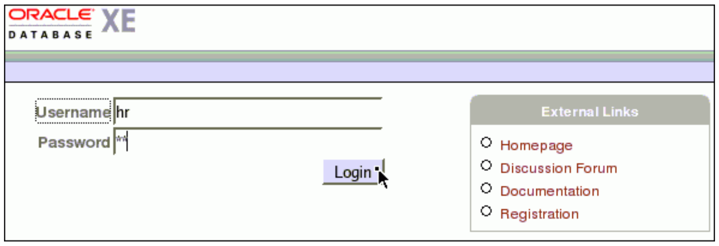
1. Create the `chap7` directory and copy the application files from `chap6`:

```
mkdir $HOME/public_html/chap7
cp $HOME/public_html/chap6/* $HOME/public_html/chap7
cd $HOME/public_html/chap7
```

2. In a browser, enter the URL to access the Oracle Database XE HTMLDB Web page:

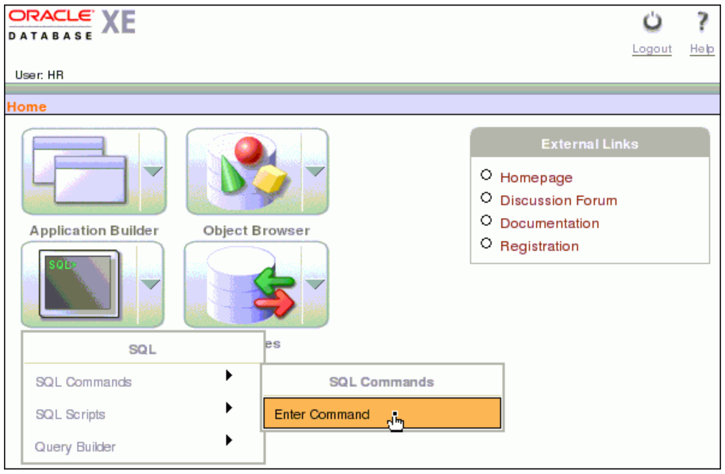
```
http://localhost:8080/htmldb
```

3. In the Oracle Database XE Login page, enter `hr` in the Username and Password fields. Click **Login**:



Username hr
Password **

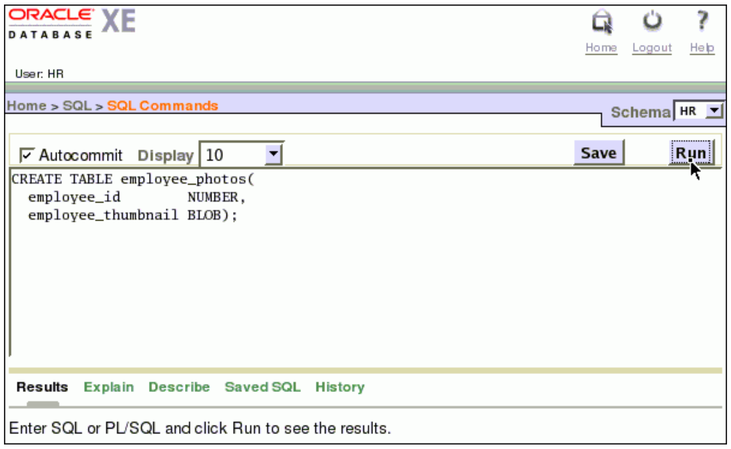
4. In the Home page, to create a new table for storing employee images, click the arrow on the **SQL** icon, highlight **SQL Commands**, and click **Enter Command**:



5. In the **SQL Commands** page, enter the following `CREATE TABLE` statement:

```
CREATE TABLE employee_photos(
  employee_id        NUMBER,
  employee_thumbnail BLOB);
```

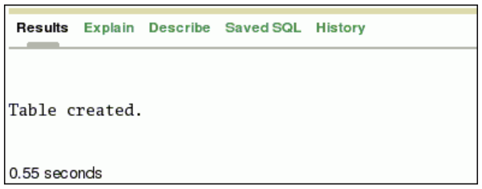
Click **Run**:



```
CREATE TABLE employee_photos(
  employee_id        NUMBER,
  employee_thumbnail BLOB);
```

Enter SQL or PL/SQL and click Run to see the results.

6. In the Results section below the command text area, confirm that the table is successfully created:



```
Table created.
```

0.55 seconds

The `HR` user must have the `CREATE TABLE` privilege to perform this command. If you get an "insufficient privileges" error message then logout as the `HR` user, login as `system` with password `manager` and execute the following `GRANT` command:

```
GRANT create table TO hr;
```

Then login as `HR` again to execute the `CREATE TABLE` command.

7. Edit `anyco_ui.inc`. Add a Photograph column to the `EMPLOYEES` table in `ui_print_employees()`:

```
<th>Commission<br>(%)</th>
<th>Remuneration</th>
<th>Photograph</th>
```

The data for the Photograph column is populated with an `<img>` tag whose `src` attribute is defined as a URL reference to a new `anyco_im.php` file, which will display the image for each employee.

8. Edit `anyco_ui.inc`. Add code in `ui_print_employees()` to generate an `<img>` tag referencing the `anyco_im.php` file with the employee identifier as a parameter:

   ```
   echo '<td align="right">'
          .htmlentities($emp['REMUNERATION']).'</td>';
   echo '<td><img src="anyco_im.php?showempphoto='.$emp['EMPLOYEE_ID']
          .'" alt="Employee photo"></td>';
   ```

9. Edit `anyco_ui.inc`. To enable images to be uploaded when a new employee is created, add an `enctype` attribute to the `<form>` tag in `ui_print_insert_employee()`:

   ```
   <form method="post" action="$posturl" enctype="multipart/form-data">
   ```

   At the bottom of the form add an upload field with input type of `file`:

   ```
   <tr>
     <td>Commission (%)</td>
     <td><input type="text" name="commpct" value="0" size="20"></td>
   </tr>
   <tr>
     <td>Photo</td>
     <td><input type="file" name="empphoto"></td>
   </tr>
   ```

10. Create `anyco_im.php`. This file accepts an employee identifier as a URL parameter, reads the thumbnail from the Photograph column for that employee, and returns the thumbnail image to be displayed:

    ```
    <?php    // anyco_im.php

    require('anyco_cn.inc');
    require('anyco_db.inc');
    construct_image();

    function construct_image()
    {
      if (!isset($_GET['showempphoto'])) {
        return;
      }

      $empid = $_GET['showempphoto'];

      $conn = db_connect($err);

      if (!$conn) {
        return;
      }

      $query =
        'SELECT employee_thumbnail
         FROM   employee_photos
    ```

```
       WHERE  employee_id = :eid';

    $stid = oci_parse($conn, $query);
    $r = oci_bind_by_name($stid, ":eid", $empid, -1);
    if (!$r) {
      return;
    }
    $r = oci_execute($stid, OCI_DEFAULT);
    if (!$r) {
      return;
    }

    $arr = oci_fetch_row($stid);
    if (!$arr) {
      return;                     // photo not found
    }

    $result = $arr[0]->load();

    // If any text (or whitespace!) is printed before this header is sent,
    // the text won't be displayed. The image also won't display properly.
    // Comment out the "header" line to see the text and debug.
    header("Content-type: image/JPEG");
    echo $result;
  }

  ?>
```

The `construct_image()` function uses the `OCI-Lob->load()` function to retrieve the Oracle LOB data, which is the image data. The PHP `header()` function sets the MIME type in the HTTP response header to ensure the browser interprets the data as a JPEG image.

11. Edit `anyco_db.inc`. Add a new function `db_insert_thumbnail()` to insert an image into the `EMPLOYEE_PHOTOS` table:

```
  function db_insert_thumbnail($conn, $empid, $imgfile, &$e)
  {
    $lob = oci_new_descriptor($conn, OCI_D_LOB);
    if (!$lob) {
      $e = db_error($conn, __FILE__, __LINE__);
      return false;
    }

    $insstmt =
      'INSERT INTO employee_photos (employee_id, employee_thumbnail)
       VALUES(:eid, empty_blob())
       RETURNING employee_thumbnail into :etn';

    $stmt = oci_parse($conn, $insstmt);
    $r = oci_bind_by_name($stmt, ':etn', $lob, -1, OCI_B_BLOB);
    if (!$r) {
      $e = db_error($stid, __FILE__, __LINE__);
      return false;
    }
    $r = oci_bind_by_name($stmt, ':eid', $empid, -1);
    if (!$r) {
      $e = db_error($stid, __FILE__, __LINE__);
      return false;
    }
    $r = oci_execute($stmt, OCI_DEFAULT);
    if (!$r) {
      $e = db_error($stid, __FILE__, __LINE__);
      return false;
    }

    if (!$lob->savefile($imgfile)) {
      $e = db_error($stid, __FILE__, __LINE__);
      return false;
    }
    $lob->free();

    return true;
  }
```

To tie the new EMPLOYEE_PHOTOS and EMPLOYEES tables together we need to use the same employee id in both.

12. Edit anyco_db.inc. Change the $bindvars parameter in db_execute_statement() to &$bindvars so that OUT bind variable values are returned from the database. At the bottom of the function add a loop to set any return bind values:

```
  function db_execute_statement($conn, $statement, &$e, &$bindvars = array())
  {
    ...

    $r = @oci_execute($stid);
    if (!$r) {
      $e = db_error($stid, __FILE__, __LINE__);
      return false;
    }
    $outbinds = array();
    foreach ($bindvars as $b) {
      $outbinds[$b[0]] = $$b[0];
    }
    $bindvars = $outbinds;
    return true;
  }
```

13. Edit anyco.php. Change the INSERT statement in insert_new_emp() so that it returns the new employee identifier in the bind variable :neweid. This value is inserted with the image into the new EMPLOYEE_PHOTOS table.

```
$statement =
  'INSERT INTO employees
              (employee_id, first_name, last_name, email, hire_date,
              job_id, salary, commission_pct, department_id)
   VALUES (employees_seq.nextval, :fnm, :lnm, :eml, :hdt,
          :jid, :sal, :cpt, :did)
   RETURNING employee_id into :neweid';
```

Also in insert_new_emp(), add an array_push() to set a new bind variable NEWEID at the end of the list of array_push() calls:

```
  array_push($bindargs, array('CPT', $newemp['commpct'], -1));
  array_push($bindargs, array('DID', $newemp['deptid'], -1));
  array_push($bindargs, array('NEWEID', null, 10));
```

Because the value of NEWID is being retrieved with the RETURNING clause in the INSERT statement, its initial value is set to NULL . The length is set to 10 to allow enough digits in the return value.

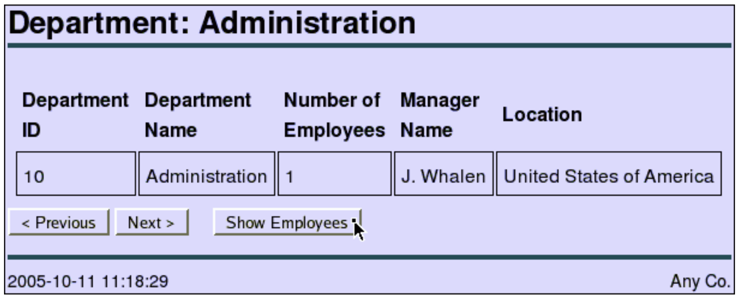
14. Edit `anyco.php`. In `insert_new_emp()`, add a call between the `db_execute_statement()` and `construct_employees()` calls to insert the thumbnail picture:

```
$r = db_execute_statement($conn, $statement, $err, $bindargs);
if ($r) {
  $r = db_insert_thumbnail($conn, $bindargs['NEWEID'],
                           $_FILES['empphoto']['tmp_name'], $e);
  construct_employees();
}
```

15. In a browser, enter the following application URL:

```
http://localhost/~<username>/chap7/anyco.php
```

16. In the Departments page, click **Show Employees** to navigate to the Employees page:

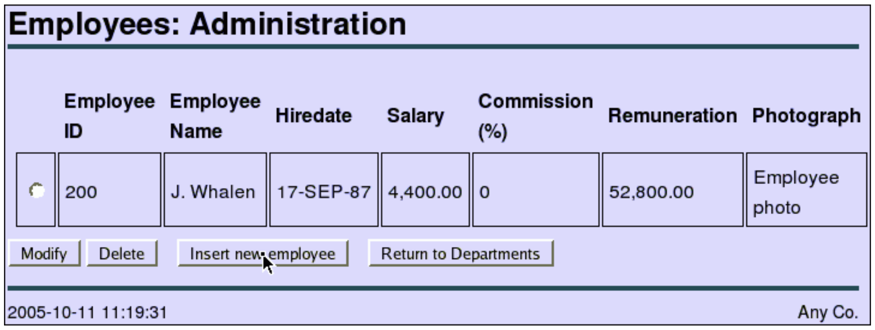


Department: Administration

Department ID	Department Name	Number of Employees	Manager Name	Location
10	Administration	1	J. Whalen	United States of America

2005-10-11 11:18:29 Any Co.

17. In the Employees page, to insert a new employee click **Insert new employee**:

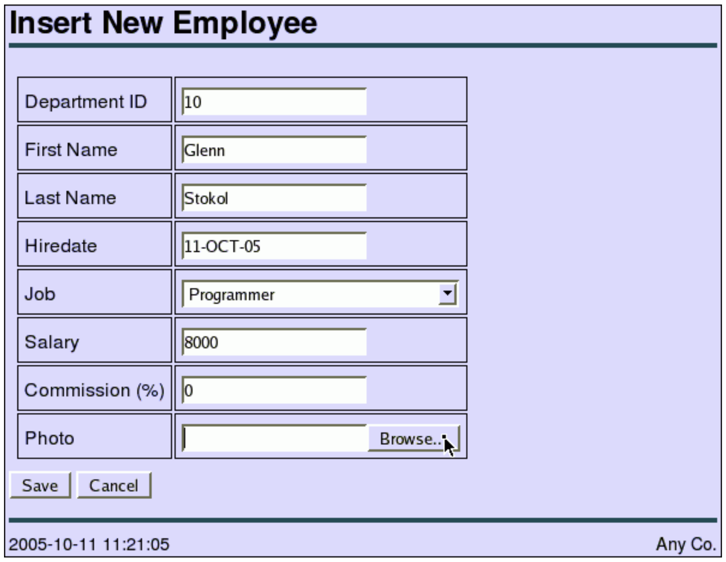


Employees: Administration

	Employee ID	Employee Name	Hiredate	Salary	Commission (%)	Remuneration	Photograph
○	200	J. Whalen	17-SEP-87	4,400.00	0	52,800.00	Employee photo

2005-10-11 11:19:31 Any Co.

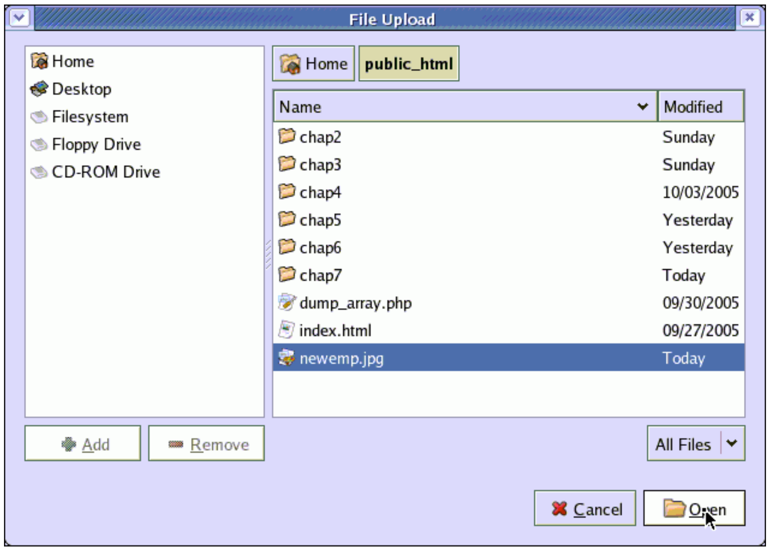
18. The Insert New Employee form allows you to choose a thumbnail image on your system to be uploaded to the database. Enter your own values in the fields or use the values as shown. Click **Browse**:


Insert New Employee

Department ID	10
First Name	Glenn
Last Name	Stokol
Hiredate	11-OCT-05
Job	Programmer
Salary	8000
Commission (%)	0
Photo	

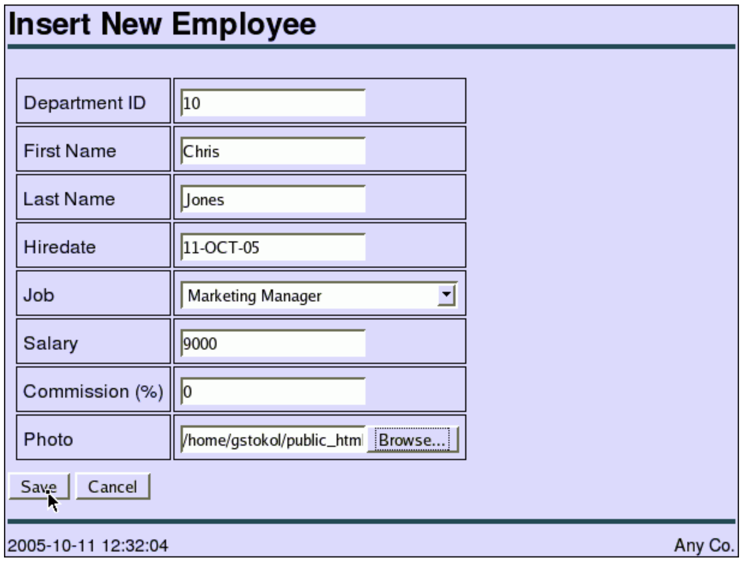
2005-10-11 11:21:05 Any Co.

19. In the File Upload window, browser for and select an image file. Click **Open**:



Name	Modified
chap2	Sunday
chap3	Sunday
chap4	10/03/2005
chap5	Yesterday
chap6	Yesterday
chap7	Today
dump_array.php	09/30/2005
index.html	09/27/2005
newemp.jpg	Today

20. In the Insert New Employee page, click **Save**:

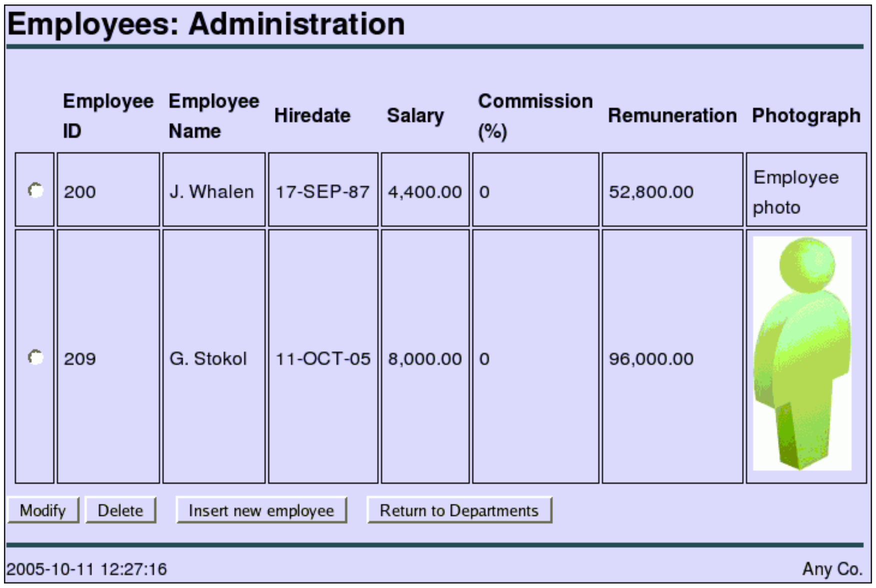


Insert New Employee

Department ID	10
First Name	Chris
Last Name	Jones
Hiredate	11-OCT-05
Job	Marketing Manager
Salary	9000
Commission (%)	0
Photo	/home/gstokol/public_html

2005-10-11 12:32:04 Any Co.

On success, the Employees page is displayed with the new employee including the image, which is displayed at its original size:



Employees: Administration

	Employee ID	Employee Name	Hiredate	Salary	Commission (%)	Remuneration	Photograph
○	200	J. Whalen	17-SEP-87	4,400.00	0	52,800.00	Employee photo
○	209	G. Stokol	11-OCT-05	8,000.00	0	96,000.00	

2005-10-11 12:27:16 Any Co.

Resizing Images

The Employee thumbnails can be resized with PHP's GD graphicsextension.

1. To turn on the graphic extension, enter the following URL in your browser to access the Zend Core for Oracle Console:

```
http://localhost/ZendCore
```

2. At the login screen, in the Password field enter the password you provided when Zend Core for Oracle was installed, and click the login (>>>) icon.
3. In the Console page, click the **Configuration** tab.
4. In the Configuration tab page, click the **Extensions** sub-tab.

5. In the Extension sub-tab page, expand the Zend Core Extensions tree control. Locate the **gd -GD (Image Manipulation)** entry and change its switch to `on` or `enabled`.
6. In the Extension sub-tab page, to save the configuration changes click the **Save Setting** link.
7. In the Extension sub-tab page, to restart the web server click the **Restart Server** link.
8. To logout of the Zend Core for Oracle Console, click the **Logout** link.
9. Edit `anyco_db.inc`. To resize the image if it is larger than a thumbnail, add the following code before the call to `$lob->savefile($imgfile)` in `db_insert_thumbnail()`:

```
$r = oci_execute($stmt, OCI_DEFAULT);
if (!$r) {
  $e = db_error($stid, __FILE__, __LINE__);
  return false;
}

// Resize the image to a thumbnail
define('MAX_THUMBNAIL_DIMENSION', 100);
$src_img = imagecreatefromjpeg($imgfile);
list($w, $h) = getimagesize($imgfile);
if ($w > MAX_THUMBNAIL_DIMENSION || $h > MAX_THUMBNAIL_DIMENSION)
{
  $scale =  MAX_THUMBNAIL_DIMENSION / (($h > $w) ? $h : $w);
  $nw = $w * $scale;
  $nh = $h * $scale;

  $dest_img = imagecreatetruecolor($nw, $nh);
  imagecopyresampled($dest_img, $src_img, 0, 0, 0, 0, $nw, $nh, $w, $h);

  imagejpeg($dest_img, $imgfile);  // overwrite file with new thumbnail

  imagedestroy($src_img);
  imagedestroy($dest_img);
}

if (!$lob->savefile($imgfile)) {
...
```

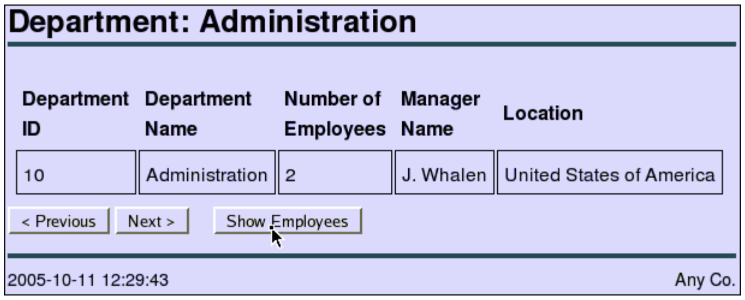
   The `imagecreatefromjpeg()` function reads the JPEG file and creates an internal representation used by subsequent GD functions. Next, new dimensions are calculated with the longest side no larger than 100 pixels. A template image with the new size is created using `imagecreatetruecolor()`. Data from the original image is sampled into it with `imagecopyresampled()` to create the thumbnail. The thumbnail is written back to the original file and the internal representations of the images are freed.

   The existing code in `db_insert_thumbnail()` uploads the image file to the database as it did in the previous section.

10. Enter the following URL in your browser to test the changes in your application:

```
http://localhost/~<username>/chap7/anyco.php
```

11. In the Departments page, navigate to the employees page by clicking **Show Employees**:

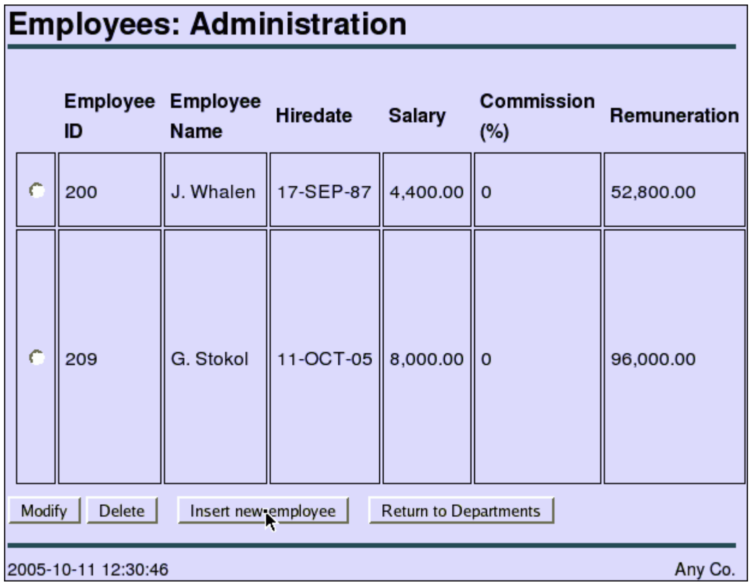


Department: Administration

Department ID	Department Name	Number of Employees	Manager Name	Location
10	Administration	2	J. Whalen	United States of America

2005-10-11 12:29:43 Any Co.

12. In the Employees page, to insert a new employee click **Insert new employee**:

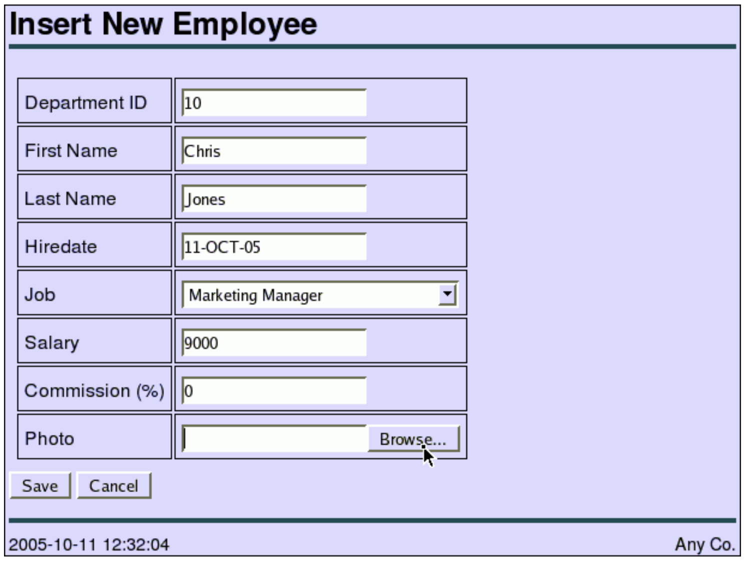


Employees: Administration

	Employee ID	Employee Name	Hiredate	Salary	Commission (%)	Remuneration
○	200	J. Whalen	17-SEP-87	4,400.00	0	52,800.00
○	209	G. Stokol	11-OCT-05	8,000.00	0	96,000.00

2005-10-11 12:30:46 Any Co.

13. Enter your new employee details or use the values shown. To browse for an employee image click **Browse**:

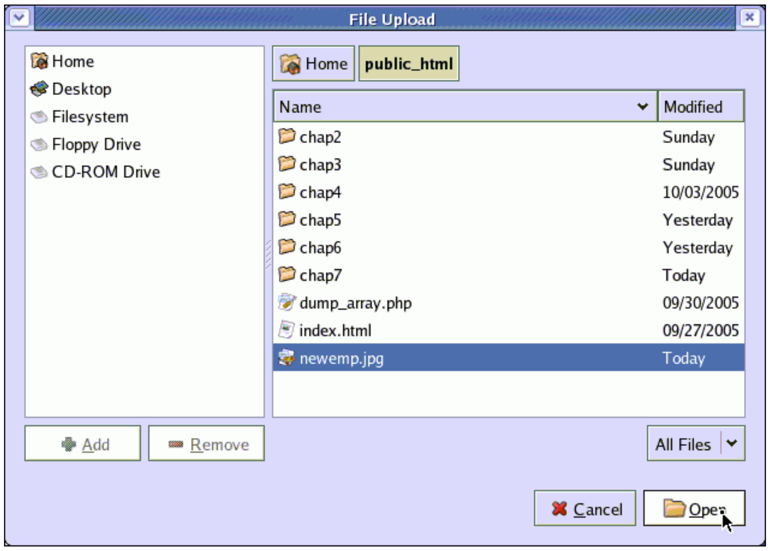


Insert New Employee

Department ID	10
First Name	Chris
Last Name	Jones
Hiredate	11-OCT-05
Job	Marketing Manager
Salary	9000
Commission (%)	0
Photo	Browse...

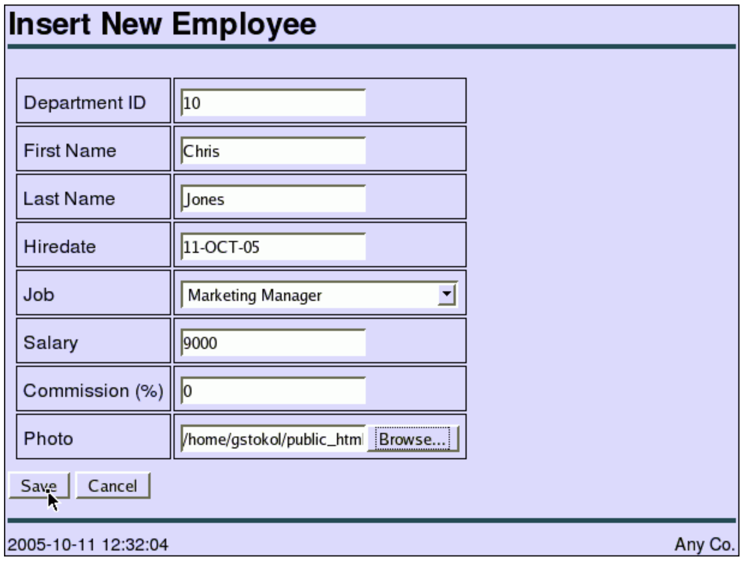
2005-10-11 12:32:04 Any Co.

14. Locate and select an image with a size larger than 100 pixels. Click **Open**:



Name	Modified
chap2	Sunday
chap3	Sunday
chap4	10/03/2005
chap5	Yesterday
chap6	Yesterday
chap7	Today
dump_array.php	09/30/2005
index.html	09/27/2005
newemp.jpg	Today

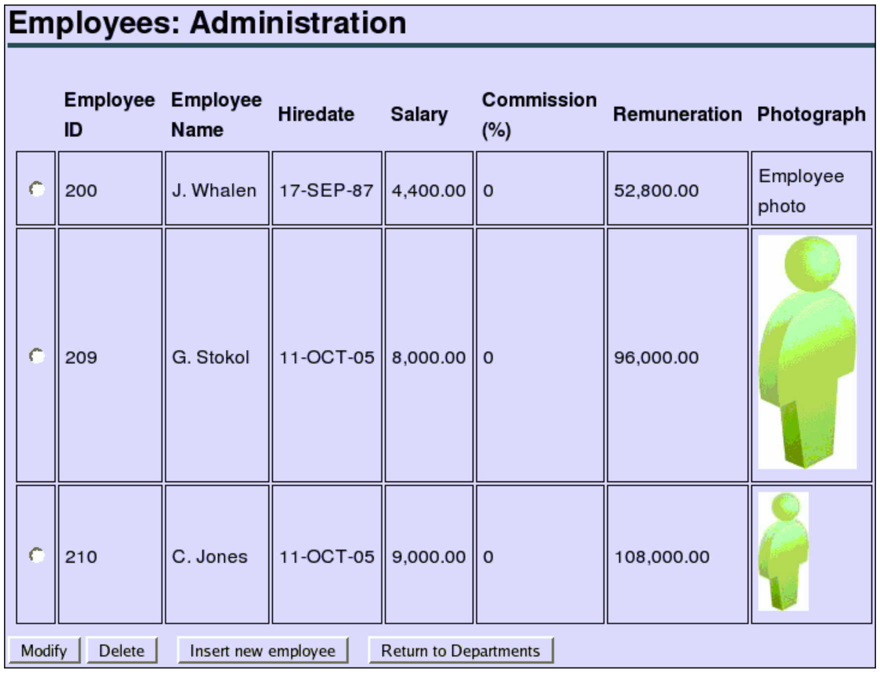
15. In the Insert New Image page, click **Save**:


Insert New Employee

Department ID	10
First Name	Chris
Last Name	Jones
Hiredate	11-OCT-05
Job	Marketing Manager
Salary	9000
Commission (%)	0
Photo	/home/gstokol/public_html

2005-10-11 12:32:04 Any Co.

The Employees page shows the new uploaded JPEG image with a reduced image size, compared to the image loaded prior to the image resize code being included:



Employees: Administration

	Employee ID	Employee Name	Hiredate	Salary	Commission (%)	Remuneration	Photograph
○	200	J. Whalen	17-SEP-87	4,400.00	0	52,800.00	Employee photo
○	209	G. Stokol	11-OCT-05	8,000.00	0	96,000.00	
○	210	C. Jones	11-OCT-05	9,000.00	0	108,000.00	

8 Building Global Applications

This chapter discusses global application development in a PHP and Oracle Database Express environment. It addresses the basic tasks associated with developing and deploying global Internet applications, including developing locale awareness, constructing HTML content in the user preferred language, and presenting data following the cultural conventions of the user's locale.

Building a global Internet application that supports different locales requires good development practices. A locale refers to a national language and the region in which the language is spoken. The application itself must be aware of the user's locale preference and be able to present content following the cultural convention expected by the user. It is important to present data with appropriate locale characteristics, such as using the correct date and number formats. Oracle Database Express is fully internationalized to provide a global platform for developing and deploying global applications.

This chapter has the following topics:

Establishing the Environment between Oracle and PHP

Correctly setting up the connectivity between the PHP engine and the Oracle database is first step in building a global application, it guarantees data integrity across all tiers. Most internet based standards support Unicode as a character encoding, in this chapter we will focus on using Unicode as the character set for data exchange.

PHP is an Oracle OCI application and rules that apply to OCI also apply to PHP. Oracle locale behavior including the client character set used in OCI application are defined by an environment variable `NLS_LANG` which has the form:

```
<language>_<territory>.<character set>
```

For example, for a German user in Germany running their application in Unicode, `NLS_LANG` should be set to

```
GERMAN_GERMANY.AL32UTF8
```

The language and territory settings control Oracle behaviors such as the Oracle date format, error message language and the rules used for sort order. The character set AL32UTF8 is Oracle's name for UTF-8.

> **See also:** Oracle Database Express Edition installation guides for information on the `NLS_LANG` environment variable.

When Zend Core for Oracle is installed on Apache, `NLS_LANG` can be set in `/etc/profile`:

```
export NLS_LANG GERMAN_GERMANY.AL32UTF8
```

If Zend Core for Oracle is installed on Oracle HTTP Server, `NLS_LANG` needs to be set as an environment variable in `$ORACLE_HOME/opmn/conf/opmn.xml`:

```
<ias-component id="HTTP_Server">
  <process-type id="HTTP_Server" module-id="OHS">
    <environment>
      <variable id="PERL5LIB"
       value="D:\oracle\1012J2EE\Apache\Apache\mod_perl\site\5.6.1\lib"/>
      <variable id="PHPRC" value="D:\oracle\1012J2EE\Apache\Apache\conf"/>
      <variable id="NLS_LANG" value="german_germany.al32utf8"/>
    </environment>
    <module-data>
      <category id="start-parameters">
        <data id="start-mode" value="ssl-disabled"/>
      </category>
    </module-data>
    <process-set id="HTTP_Server" numprocs="1"/>
  </process-type>
</ias-component>
```

The web listener will need to be restarted to pick up the change.

String Manipulation

PHP was designed to work with the ISO-8859-1 character set. To handle other character sets, specifically multi-byte character sets, a set of "Multi-Byte String Functions" is available. To enable these functions open the Zend Core for Oracle console and go to the Configuration tab.

Navigate to the Extensions sub-tab and expand the Zend Core Extensions tree control.

Your application code should use functions such as `mb_strlen()` to calculate the number of characters in strings. This may return different value than `strlen()`, which will return the number of bytes in a string.

Once the mbstring extension has been enabled and the web server restarted, several configuration options become available. You can change the behavior of the standard PHP string functions by setting `mbstring.func_overload` to one of the "Overload" settings.

The PHP mbstring reference manual `http://www.php.net/mbstring` contains more information.

Determining User's Locale

In a global environment, your application will need to accept users with different locale preferences. The application need to determine the user's preferred locale. Once that is known, the application should construct HTML content in the language of the locale, and follows the cultural conventions implied by the locale.

One of the most common methods in determining a user's locale, is based on the default ISO locale setting of the user's browser. Every HTTP request sends the default ISO locale of the browser, with the Accept-Language HTTP header. If the Accept-Language header is NULL, then the locale should default to English.

The following PHP code will retrieve the ISO locale from the Accept-Language HTTP header via the `$_SERVER` Server variable.

```
$s = $_SERVER["HTTP_ACCEPT_LANGUAGE"]
```

Developing Locale Awareness

Once the user's locale preference has been determined, the application can call locale-sensitive functions, such as date, time, and monetary formatting to format the HTML pages according to the cultural conventions of the user's locale.

When writing global applications across different programming environment, the user locale settings must be synchronized between environments. For example, PHP applications that call PL/SQL procedures should map the ISO locales to the corresponding `NLS_LANGUAGE` and `NLS_TERRITORY` values and change the parameter values to match the user's locale before calling the PL/SQL procedures. The table below shows how some of the commonly used locales are defined in ISO and Oracle environ-ments.

Table 8–1 Locale Representations in ISO, SQL and PL/SQL Programming Environments

Locale	Locale ID	NLS_LANGUAGE	NLS_TERRITORY
Chinese (R.P.C.)	zh-CN	SIMPLIFIED CHINESE	CHINA
Chinese (Taiwan)	zh-TW	TRADITIONAL CHINESE	TAIWAN
English (U.S.A)	en-US	AMERICAN	AMERICA
English (United Kingdom)	en-GB	ENGLISH	UNITED KINGDOM
French (Canada)	fr-CA	CANADIAN FRENCH	CANADA
French (France)	fr-FR	FRENCH	FRANCE
German	de	GERMAN	GERMANY
Italian	it	ITALIAN	ITALY
Japanese	ja	JAPANESE	JAPAN
Korean	ko	KOREAN	KOREA
Portuguese (Brazil)	pt-BR	BRAZILIAN PORTUGUESE	BRAZIL

Table 8–1 (Cont.) Locale Representations in ISO, SQL and PL/SQL Programming

Locale	Locale ID	NLS_LANGUAGE	NLS_TERRITORY
Portuguese	pt	PORTUGUESE	PORTUGAL
Spanish	es	SPANISH	SPAIN

Encoding HTML Pages

The encoding of an HTML page is important information for a browser and an Internet application. You can think of the page encoding as the character set used for the locale that an Internet application is serving. The browser needs to know about the page encoding so that it can use the correct fonts and character set mapping tables to display the HTML pages. Internet applications need to know about the HTML page encoding so they can process input data from an HTML form.

Instead of using different native encodings for the different locales, it is recommended to use UTF-8 (Unicode encoding) for all page encodings. Using the UTF-8 encoding not only simplifies the coding for global applications, but it allows for multilingual content on a single page.

Specifying the Page Encoding for HTML Pages

There are two ways to specify the encoding of an HTML page, one is in the HTTP header, and the other is in the HTML page header.

Specifying the Encoding in the HTTP Header

Include the Content-Type HTTP header in the HTTP specification. It specifies the content type and character set. The Content-Type HTTP header has the following form:

```
Content-Type: text/html; charset=utf-8
```

The charset parameter specifies the encoding for the HTML page. The possible values for the charset parameter are the IANA names for the character encodings that the browser supports.

Specifying the Encoding in the HTML Page Header

Use this method primarily for static HTML pages. Specify the character encoding in the HTML header as follows:

```
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
```

The charset parameter specifies the encoding for the HTML page. As with the Content-Type HTTP Header, the possible values for the charset parameter are the IANA names for the character encodings that the browser supports.

Specifying the Page Encoding in PHP

You can specify the encoding of an HTML page in the Content-Type HTTP header in PHP by setting the `default_charset` configuration variable as follows:

```
default_charset = UTF-8
```

This can be found in the Zend Core for Oracle Console in the Configuration tab. Choose the PHP sub-tab and expand the Data Handling tree control. After entering a value, save the configuration settings and restart the web server.

This setting does not imply any conversion of outgoing pages. Your application must ensure the server generated pages are encoded in UTF-8.

Organizing the Content of HTML Pages for Translation

Making the user interface available in the user's local language is one of the fundamental task in globalizing an application. Translatable sources for the content of an HTML page belong to the following categories:

- Text strings hard-coded in the application code
- Static HTML files, images files, and template files such as CSS
- Dynamic data stored in the database

Strings in PHP

You should externalize translatable strings within your PHP application logic, so that the text can be easily available for translation. These text messages can be stored in flat files or database tables depending on the type and the volume of the data being translated.

Static Files

Static files such as HTML and GIF files are readily translatable. When these files are translated, they should be translated into the corresponding language with UTF8 as the file encoding. To differentiate the languages of the translated files, the static files of different languages can be staged in different directories or with different file names.

Data from the Database

Dynamic information such as product names and product descriptions are most likely stored in the database. In order to differentiate various translations, the database schema holding these information should include a column to indicate the language of the information. To select the translated information, you need to include the WHERE clause in your query to select the information in the desired language of the query.

Presenting Data following User's Locale Convention

Data in the application needs to be presented in a way that conforms to the user's expectation, if not, the meaning of the data can sometimes be mis-interpreted. For example, the date '12/11/05' implies '11th December 2005' in the United States, whereas in the United Kingdom it means '12th November 2005'. Similar confusion exists for number and monetary formats, the symbol dot '.' is a decimal separator in the United States, in Germany this symbol is recognized as a thousand separator.

Different languages have their own sorting rules, some languages are collated according to the letter sequence in the alphabet, some according to the number of stroke counts in the letter, and there are some languages which are ordered by the pronunciation of the words. Presenting data not sorted in the linguistic sequence that your users are accustomed to can make searching for information difficult and time consuming.

Oracle Database Express offers many features that help to refine the presentation of data when the user's locale preference is known. Here are some examples of locale sensitive operations in SQL.

Oracle Date Formats

There are three different date presentation formats in Oracle Database Express Edition, they are standard, short, and long dates. The examples below, illustrate the differences between the short data and long date formats for both United States and Germany.

```
SQL> alter session set nls_territory=america nls_language=american;

Session altered.


SQL> select employee_id EmpID,
  2  substr(first_name,1,1)||'.'||last_name "EmpName",
  3  to_char(hire_date,'DS') "Hiredate",
  4  to_char(hire_date,'DL') "Long HireDate"
  5  from employees
  6* where employee_id <105;

     EMPID EmpName                    Hiredate   Long HireDate
---------- -------------------------- ---------- -----------------------------
       100 S.King                     06/17/1987 Wednesday, June 17, 1987
       101 N.Kochhar                  09/21/1989 Thursday, September 21, 1989
       102 L.De Haan                  01/13/1993 Wednesday, January 13, 1993
       103 A.Hunold                   01/03/1990 Wednesday, January 3, 1990
       104 B.Ernst                    05/21/1991 Tuesday, May 21, 1991

SQL> alter session set nls_territory=germany nls_language=german;

Session altered.

SQL> select employee_id EmpID,
  2  substr(first_name,1,1)||'.'||last_name "EmpName",
  3  to_char(hire_date,'DS') "Hiredate",
  4  to_char(hire_date,'DL') "Long HireDate"
  5  from employees
  6* where employee_id <105;

     EMPID EmpName                    Hiredate Long HireDate
---------- -------------------------- -------- ------------------------------
       100 S.King                     17.06.87 Mittwoch, 17. Juni 1987
       101 N.Kochhar                  21.09.89 Donnerstag, 21. September 1989
       102 L.De Haan                  13.01.93 Mittwoch, 13. Januar 1993
       103 A.Hunold                   03.01.90 Mittwoch, 3. Januar 1990
       104 B.Ernst                    21.05.91 Dienstag, 21. Mai 1991
```

Oracle Number Formats

The examples below, illustrate the differences in the decimal character and group separator between United States and Germany.

```
SQL> alter session set nls_territory=america;

Session altered.

SQL> select employee_id EmpID,
  2  substr(first_name,1,1)||'.'||last_name "EmpName",
  3  to_char(salary, '99G999D99') "Salary"
  4  from employees
  5* where employee_id <105
```

```
     EMPID EmpName                     Salary
---------- --------------------------- ----------
       100 S.King                       24,000.00
       101 N.Kochhar                    17,000.00
       102 L.De Haan                    17,000.00
       103 A.Hunold                      9,000.00
       104 B.Ernst                       6,000.00

SQL> alter session set nls_territory=germany;

Session altered.

SQL> select employee_id EmpID,
  2  substr(first_name,1,1)||'.'||last_name "EmpName",
  3  to_char(salary, '99G999D99') "Salary"
  4  from employees
  5* where employee_id <105

     EMPID EmpName                     Salary
---------- --------------------------- ----------
       100 S.King                       24.000,00
       101 N.Kochhar                    17.000,00
       102 L.De Haan                    17.000,00
       103 A.Hunold                      9.000,00
       104 B.Ernst                       6.000,00
```

Oracle Linguistic Sorts

Spain traditionally treats *ch*, *ll* as well as *ñ* as letters of their own, ordered after *c*, *l* and *n* respectively. The examples below, illustrate the effect of using a Spanish sort against the employee names Chen and Chung.

```
SQL> alter session set nls_sort=binary;

Session altered.

SQL> select employee_id EmpID,
  2         last_name "Last Name"
  3  from employees
  4  where last_name like 'C%'
  5* order by last_name

     EMPID Last Name
---------- -------------------------
       187 Cabrio
       148 Cambrault
       154 Cambrault
       110 Chen
       188 Chung
       119 Colmenares

6 rows selected.

SQL> alter session set nls_sort=spanish_m;

Session altered.

SQL> select employee_id EmpID,
  2         last_name "Last Name"
  3  from employees
  4  where last_name like 'C%'
  5* order by last_name

     EMPID Last Name
---------- -------------------------
       187 Cabrio
       148 Cambrault
       154 Cambrault
       119 Colmenares
       110 Chen
       188 Chung

6 rows selected.
```

Oracle Error Messages

The NLS_LANGUAGE parameter also controls the language of the database error messages being returned from the database. Setting this parameter prior to submitting your SQL statement will ensure that the language specific database error messages will be returned to the application.

Consider the following server message:

```
ORA-00942: table or view does not exist
```

When the NLS_LANGUAGE parameter is set to French, the server message appears as follows:

```
ORA-00942: table ou vue inexistante
```

See also: The "Working in a Global Environment" chapter in the *Oracle Database Express Edition 2 Day Developer Guide* for more discussion of globalization support features within Oracle Database Express Edition.

Index